|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Subject | QPlay QChat API | | Doc.: |  | | Rev. | V1.2.7 |
|  | | | Release Date: | 2017/11/07 | | Revision Status: | |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  | | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  |  | | | | |  |  |
|  | | | | | | | |
|  | | | | | | | |
|  | | | | | | | |
|  | | | | | | | |
|  | | | | | | | |
|  | | | | | | | |
| Approved By: | | Reviewed By: | | | Prepared By: | | |
| PM Head / | | SD / | | | PM /Steven.Yan | | |
|  | | QJ / | | |  | | |
|  | |  | | |  | | |
|  | |  | | |  | | |
|  | |  | | |  | | |
|  | |  | | |  | | |

[1.0 Change History 4](#_Toc494285070)

[2.0 Introduction 5](#_Toc494285071)

[2.1 System Description 5](#_Toc494285072)

[2.2 Purpose & Scope 5](#_Toc494285073)

[3.0 iOS Function Description: 6](#_Toc494285074)

[3.1 集成步骤 6](#_Toc494285075)

[3.2 Demo 6](#_Toc494285076)

[3.3 User 10](#_Toc494285077)

[3.4 Message 11](#_Toc494285078)

[3.5 Conversation 12](#_Toc494285079)

[3.6 Group 15](#_Toc494285080)

[3.7 FriendManager 17](#_Toc494285081)

[3.8 Disturb 21](#_Toc494285082)

[3.9 DocumentEvent 22](#_Toc494285083)

[4.0 Android Function Description: 25](#_Toc494285084)

[4.1 集成步骤 25](#_Toc494285085)

[4.2 Demo 26](#_Toc494285086)

[4.3 用户属性维护 29](#_Toc494285087)

內容

[1.0 Change History 4](#_Toc497744571)

[2.0 Introduction 6](#_Toc497744572)

[2.1 System Description 6](#_Toc497744573)

[2.2 Purpose & Scope 6](#_Toc497744574)

[3.0 iOS Function Description: 7](#_Toc497744575)

[3.1 集成步骤 7](#_Toc497744576)

[3.2 Demo 7](#_Toc497744577)

[3.3 User 11](#_Toc497744578)

[3.3.1 \*API - login 11](#_Toc497744579)

[3.3.2 API - logout 11](#_Toc497744580)

[3.3.3 API - getMyInfo 11](#_Toc497744581)

[3.3.4 API - getUserInfo 11](#_Toc497744582)

[3.3.5 API - getUserInfoArray 12](#_Toc497744583)

[3.3.6 API - updateMyInfo（与 Android 通用） 12](#_Toc497744584)

[3.4 Message 12](#_Toc497744585)

[3.4.1 API - sendGroupTextMessage 13](#_Toc497744586)

[3.4.2 API - sendGroupImageMessage 13](#_Toc497744587)

[3.5 Conversation 13](#_Toc497744588)

[3.5.1 API - getGroupConversationHistoryMessage 13](#_Toc497744589)

[3.5.2 API - getAllGroupConversation 14](#_Toc497744590)

[3.5.3 API - deleteGroupConversation 14](#_Toc497744591)

[3.5.4 API - getAllUnreadCount 14](#_Toc497744592)

[3.5.5 API - clearGroupUnreadCount 14](#_Toc497744593)

[3.5.6 \*API - sendFileMessage 15](#_Toc497744594)

[3.5.7 \*API - sendLocationMessage 15](#_Toc497744595)

[3.6 Group 16](#_Toc497744596)

[3.6.1 API - createGroupIniOS 16](#_Toc497744597)

[3.6.2 API - updateGroupInfo 16](#_Toc497744598)

[3.6.3 API - getGroupInfo 16](#_Toc497744599)

[3.6.4 API - myGroupArray 16](#_Toc497744600)

[3.6.5 API - memberArray 17](#_Toc497744601)

[3.6.6 API - addMembers 17](#_Toc497744602)

[3.6.7 API - removeMembers 17](#_Toc497744603)

[3.6.8 API - exitGroup 17](#_Toc497744604)

[3.7 FriendManager 18](#_Toc497744605)

[3.7.1 API - getQFriendList 18](#_Toc497744606)

[3.7.2 Custom API 18](#_Toc497744607)

[3.7.3 API - sendInvitationRequest 21](#_Toc497744608)

[3.7.4 API - acceptInvitation 22](#_Toc497744609)

[3.7.5 API - rejectInvitation 22](#_Toc497744610)

[3.7.6 API - removeFriend 22](#_Toc497744611)

[3.8 Disturb 22](#_Toc497744612)

[3.8.1 \*API - noDisturbList 22](#_Toc497744613)

[3.8.2 \*API - isSetGlobalNoDisturb 23](#_Toc497744614)

[3.8.3 \*API - setIsGlobalNoDisturb 23](#_Toc497744615)

[3.8.4 \*API - userSetIsNoDisturb 23](#_Toc497744616)

[3.9 DocumentEvent 23](#_Toc497744617)

[4.0 Android Function Description: 26](#_Toc497744618)

[4.1 集成步骤 26](#_Toc497744619)

[4.2 Demo 27](#_Toc497744620)

[4.2.1 \*login 29](#_Toc497744621)

[4.2.2 logout 29](#_Toc497744622)

[4.3 用户属性维护 30](#_Toc497744623)

[4.3.1 getUserInfo 30](#_Toc497744624)

[4.3.2 getMyInfo 31](#_Toc497744625)

[4.3.3 updateMyInfo 31](#_Toc497744626)

[4.3.4 updateMyAvatar 32](#_Toc497744627)

[4.3.5 getUserAvatar 33](#_Toc497744628)

[4.3.6 getOriginalUserAvatar 33](#_Toc497744629)

[4.4 发送消息 34](#_Toc497744630)

[4.4.1 sendGroupTextMessage 34](#_Toc497744631)

[4.4.2 sendGroupImageMessage 35](#_Toc497744632)

[4.4.3 \*sendGroupVoiceMessage 35](#_Toc497744633)

[4.4.4 \*sendGroupLocationMessage 36](#_Toc497744634)

[4.4.5 \*sendGroupFileMessage 36](#_Toc497744635)

[4.5 获取历史消息 37](#_Toc497744636)

[4.5.1 getOfflineMessages 37](#_Toc497744637)

[4.5.2 getLatestMessage 37](#_Toc497744638)

[4.5.3 getHistoryMessages 38](#_Toc497744639)

[4.5.4 getAllMessages 39](#_Toc497744640)

[4.5.5 getOriginImageInGroupConversation 40](#_Toc497744641)

[4.6 好友关系 41](#_Toc497744642)

[4.6.1 sendInvitationRequest 41](#_Toc497744643)

[4.6.2 acceptInvitation 41](#_Toc497744644)

[4.6.3 declineInvitation 42](#_Toc497744645)

[4.6.4 getFriendList 42](#_Toc497744646)

[4.6.5 removeFromFriendList 43](#_Toc497744647)

[4.6.6 jmessage.onInviteReceived 43](#_Toc497744648)

[4.7 聊天会话 43](#_Toc497744649)

[4.7.1 isGroupConversationExist 44](#_Toc497744650)

[4.7.2 getConversationList 44](#_Toc497744651)

[4.7.3 exitConversation 45](#_Toc497744652)

[4.7.4 群聊 45](#_Toc497744653)

[4.8 群组 48](#_Toc497744654)

[4.8.1 createGroup 48](#_Toc497744655)

[4.8.2 getGroupIDList 49](#_Toc497744656)

[4.8.3 getGroupInfo 50](#_Toc497744657)

[4.8.4 updateGroupName 50](#_Toc497744658)

[4.8.5 \*updateGroupDescription 51](#_Toc497744659)

[4.8.6 addGroupMembers 51](#_Toc497744660)

[4.8.7 removeGroupMembers 52](#_Toc497744661)

[4.8.8 exitGroup 53](#_Toc497744662)

[4.8.9 getGroupMembers 53](#_Toc497744663)

[4.9 事件处理 54](#_Toc497744664)

[4.9.1 消息事件 54](#_Toc497744665)

[4.9.2 用户状态变更事件 58](#_Toc497744666)

[4.9.3 群组事件 58](#_Toc497744667)

[4.9.4 好友事件 59](#_Toc497744668)

[5.0 QChat Custom API 60](#_Toc497744669)

[5.1 好友管理 60](#_Toc497744670)

[5.1.1 getQList 60](#_Toc497744671)

[5.1.2 getQFriend 64](#_Toc497744672)

[5.1.3 setQFriend 69](#_Toc497744673)

[5.1.4 sendQInvitation 72](#_Toc497744674)

[5.1.5 sendQInstall 75](#_Toc497744675)

[5.1.6 getQUserDetail 77](#_Toc497744676)

[5.1.7 setQUserDetail 81](#_Toc497744677)

[~~5.1.8~~ ~~setQUserAvator~~ (不使用) 84](#_Toc497744678)

[5.1.9 removeQFriend 86](#_Toc497744679)

[5.1.10 acceptQInvitation 89](#_Toc497744680)

[5.1.11 rejectQInvitation 92](#_Toc497744681)

[5.2 聊天室管理 95](#_Toc497744682)

[5.2.1 newQChatroom 95](#_Toc497744683)

[5.2.2 setQChatroom 101](#_Toc497744684)

[5.2.3 addQMember 105](#_Toc497744685)

[5.2.4 removeQMember 110](#_Toc497744686)

[5.2.5 getQGroupHistoryMessage 115](#_Toc497744687)

[5.2.6 getQGroupHistoryMessageJob (後台JOB專用) 120](#_Toc497744688)

[5.2.7 getQUserChatroom 135](#_Toc497744689)

[5.3 推播Token註冊 138](#_Toc497744690)

[5.3.1 sendPushToken 138](#_Toc497744691)

[6.0 JMessage Phone Gap v3.0.0 141](#_Toc497744692)

[6.1 參數設定 141](#_Toc497744693)

[6.2 登入: 143](#_Toc497744694)

[6.3 登出: 143](#_Toc497744695)

[6.4 獲取自己帳號訊息: 143](#_Toc497744696)

[6.5 獲取用戶訊息: 144](#_Toc497744697)

[6.6 送出文字訊息: 144](#_Toc497744698)

[6.7 送出圖片訊息: 144](#_Toc497744699)

[6.8 獲取歷史訊息: 145](#_Toc497744700)

[6.9 獲取當前USER所有的聊天室ID: 146](#_Toc497744701)

[6.10 獲取聊天室基本訊息: 146](#_Toc497744702)

[6.11 退出聊天室: 146](#_Toc497744703)

[6.12 獲取聊天室成員: 147](#_Toc497744704)

[6.13 下載指定消息的原圖: 147](#_Toc497744705)

[6.14 進入聊天對話: 147](#_Toc497744706)

[6.15 離開聊天對話: 148](#_Toc497744707)

[6.16 獲取單一個聊天對話: 148](#_Toc497744708)

[6.17 獲取所有聊天對話: 149](#_Toc497744709)

[6.18 重置單一個聊天對話的未讀消息數 149](#_Toc497744710)

[6.19 監聽訊息類的事件: 149](#_Toc497744711)

[6.20 監聽事件類的事件: 150](#_Toc497744712)

[6.21 監聽離線消息: 151](#_Toc497744713)

[6.22 監聽登入狀態: 151](#_Toc497744714)

[6.23 返回物件的元素組成: 152](#_Toc497744715)

[6.24 Message元素 153](#_Toc497744716)

[6.25 Event元素: 154](#_Toc497744717)

[7.0 Error Code List 155](#_Toc497744718)

[8.0 Others 157](#_Toc497744719)

[4.5 获取历史消息 36](#_Toc494285089)

[4.6 好友关系 40](#_Toc494285090)

[4.7 聊天会话 42](#_Toc494285091)

[4.8 群组 47](#_Toc494285092)

[4.9 事件处理 53](#_Toc494285093)

[5.0 QChat Custom API 59](#_Toc494285094)

[5.1 好友管理 59](#_Toc494285095)

[5.2 聊天室管理 93](#_Toc494285096)

[5.3 推播Token註冊 133](#_Toc494285097)

[6.0 JMessage Phone Gap v3.0.0 137](#_Toc494285098)

[6.1 參數設定 137](#_Toc494285099)

[6.2 登入: 139](#_Toc494285100)

[6.3 登出: 139](#_Toc494285101)

[6.4 獲取自己帳號訊息: 139](#_Toc494285102)

[6.5 獲取用戶訊息: 140](#_Toc494285103)

[6.6 送出文字訊息: 140](#_Toc494285104)

[6.7 送出圖片訊息: 140](#_Toc494285105)

[6.8 獲取歷史訊息: 141](#_Toc494285106)

[6.9 獲取當前USER所有的聊天室ID: 142](#_Toc494285107)

[6.10 獲取聊天室基本訊息: 142](#_Toc494285108)

[6.11 退出聊天室: 142](#_Toc494285109)

[6.12 獲取聊天室成員: 143](#_Toc494285110)

[6.13 下載指定消息的原圖: 143](#_Toc494285111)

[6.14 進入聊天對話: 143](#_Toc494285112)

[6.15 離開聊天對話: 144](#_Toc494285113)

[6.16 獲取單一個聊天對話: 144](#_Toc494285114)

[6.17 獲取所有聊天對話: 145](#_Toc494285115)

[6.18 重置單一個聊天對話的未讀消息數 145](#_Toc494285116)

[6.19 監聽訊息類的事件: 145](#_Toc494285117)

[6.20 監聽事件類的事件: 146](#_Toc494285118)

[6.21 監聽離線消息: 147](#_Toc494285119)

[6.22 監聽登入狀態: 147](#_Toc494285120)

[6.23 返回物件的元素組成: 148](#_Toc494285121)

[6.24 Message元素 149](#_Toc494285122)

[6.25 Event元素: 150](#_Toc494285123)

[7.0 Error Code List 151](#_Toc494285124)

[8.0 Others 153](#_Toc494285125)

# Change History

As to the revised document: the latest revised content is written in blue font, while the yellow background means the waiting confirm status. And the double strikethrough represents the deleting content.

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Comment |
| 2017-05-17 | Steven Yan | 1.0.0 | Initial |
| 2017-07-19 | Steven Yan | 1.1.0 |  |
| 2017-08-02 | Steven Yan | 1.2.0 | 加上第五章 |
| 2017-08-10 | Steven Yan | 1.2.1 | 1.補上5支API規格  2.其中5.2.6 為Jmessage的REST REPORT API V2, 並非custom API, APP端不會使用到這支  3.  其中5.2.2, 5.2.3, 5.2.4, 5.2.6 四個章節添加向JMessage查詢的REST API使用方式 |
| 2017-08-11 | Steven Yan | 1.2.2 | 1. 新增第6章節, JMessage PhoneGap v3.0.0 API  2. 5.2.5 getGroupHistoryMessage新增兩個response節點  qplay\_media\_path , qplay\_thumb\_path  3. 新增5.1.8 setQUserAvator 設定大頭照 |
| 2017-09-19 | Steven Yan | 1.2.4 | 1.新增025926錯誤碼  2.新增newChatroom去呼叫jmessage add/group的說明 |
| 2017-09-27 | Steven Yan | 1.2.5 | 新增5.3章節, sendPushToken  修改response body在文件中的表達方式  透過出現次數, 0, 1, 0-1, 0-N, 1-N等表達 |
| 2017-09-29 | StevenYan | 1.2.6 | 5.2.1newQChatroom, 增加qp\_chatroom.member欄位, 如果request\_body的chatroom\_desc內容中group\_message=N, 代表是私聊, 需要將聊天室兩人的工號寫進qp\_chatroom.member  當要新增聊天室前, 需要先檢查是否已經存在兩人的私聊, 已存在則不新建聊天室, 取舊的聊天室groupid回傳給APP  5.1.1 getQList, 新增條件, 回傳的成員中, 條件必須為qp\_user.status =’N’ |
| 2017-11-06 | CleoWChan | 1.2.7 | 5.2.10.3新增Check Point: acceptQinvitation，保護名單接受好友邀請後，會建立雙方好友關係  5.1.9.3新增Check Point : removeQFrend，一般名單為刪除我的最愛，保護名單刪除所有好友關係  5.1.2 getQFriend,增加inviter節點，顯示邀請人列表(僅回傳name,.emp\_no)  5.2.1 newQChatromm   * chatroom\_name， 聊天室標題，長度限制64byte(JMessage聊天室標題最大長度限制) * newQChatroom 後端不須推播，移除lang, need\_push,app\_key   5.1.6 setQUserDetail memo長度限制200個字  5.2.3 addQmember 後端不須推播，移除lang,need\_push,app\_key  5.2.4 removeQmember 後端不須推播，移除lang,need\_push,app\_key  5.2.5getQGroupHistoryMessage 增加sort節點，提供APP端呼叫時控制排序順序  5.2.7 新增API getQUserChatroom取得特定用戶所有聊天室資訊  7.0 Error Code List新增錯誤代碼   * 025927 已發送過交友邀請 * 025928 交友邀請不存在 * 025929 私聊聊天室不可新增成員 * 025930 調用極光API出錯 |

# Introduction

## System Description

請填入系統描述

提供QChat API資訊

## Purpose & Scope

請填入專案目的及範疇

透過API串接QChat

# iOS Function Description:

Base on Jpush官方所提供的Phonegap API實作QChat API

<https://github.com/jpush/jmessage-phonegap-plugin>

## 集成步骤

* 在线安装
* cordova plugin add jmessage-phonegap-plugin --variable APP\_KEY=your\_app\_key

或

cordova plugin add https://github.com/jpush/jmessage-phonegap-plugin.git --variable APP\_KEY=your\_app\_key

* 本地安装

cordova plugin add <Plugin Path> --variable APP\_KEY=your\_app\_key

## Demo

插件项目中的 /example 目录下包含一个简单的示例，如果想参考可以将目录下的所有文件拷贝到具体 Cordova 项目的 /assets/www/ 目录下。

API 統一說明：

* iOS 中跨應用介面均以 cross\_ 開頭，需要傳有效的 appkey，其餘方法的 appkey 參數一律傳 null
* 參數 successCallback、errorCallback 分別為成功、失敗回檔
* 參數名為 xxxArray 則傳陣列，其餘無特殊說明傳字串
* 調用示例：window.JMessage.funcName(args, successCallback, errorCallback);
* iOS API名稱列表與描述 (\*表示由QPLAY SERVER端控制)

*請列出所有會用上的Custom API, 以下為yellowpage的範例, 共會開發6支web service供QPlay使用*

|  |  |
| --- | --- |
| **Custom API名稱** | **描述** |
| \*login | 用戶登錄 |
| \*logout | 當前用戶退出登錄 |
| getMyInfo | 獲取我的資訊（當前登錄使用者） |
| getUserInfo | 獲取某使用者的資訊 |
| getUserInfoArray | 獲取使用者詳情（批量介面） |
| updateMyInfo | 獲取我的資訊（當前登錄使用者） |
| sendGroupTextMessage | 發送群聊文本消息 |
| sendGroupImageMessage | 發送群聊圖片消息 |
| getGroupConversationHistoryMessage | 獲取多條群聊消息（同步介面） |
| getAllUnreadCount | 獲取當前所有會話的未讀消息的總數 |
| clearGroupUnreadCount | 清除群聊會話未讀數 |
| \*sendFileMessage | 發送單聊、群聊檔消息 |
| \*sendLocationMessage | 發送地理位置消息 |
| \*createGroupIniOS | 創建群組 |
| updateGroupInfo | 修改群組資訊 |
| getGroupInfo | 獲取群組詳情（不包含群組成員） |
| myGroupArray | 獲取我的群組列表 |
| memberArray | 獲取當前群組成員列表 |
| \*addMembers | 添加群組成員 |
| \*removeMembers | 刪除群組成員 |
| exitGroup | 退出群組（當前用戶） |
| getFriendList | 獲取好友列表 |
| sendInvitationRequest | 發送添加好友請求 |
| acceptInvitation | 接受好友邀請 |
| rejectInvitation | 拒絕好友請求 |
| removeFriend | 刪除好友 |
| noDisturbList | 使用者免打擾清單 設置全域免打擾標識 |
| isSetGlobalNoDisturb | 獲取全域免打擾狀態 |
| setIsGlobalNoDisturb | 設置是否全域免打擾 |
| userSetIsNoDisturb | 設置用戶免打擾（支援跨應用設置） |
| groupSetIsNoDisturb | 設置群組消息免打擾 |
| ##DocumentEvent | 監聽所有事件 |

* Android API名稱列表與描述 (\*表示由QPLAY SERVER端控制)

|  |  |
| --- | --- |
| **Custom API名稱** | **描述** |
| \*login | 用戶登錄 |
| \*logout | 當前用戶退出登錄 |
| getMyInfo | 獲取我的資訊（當前登錄使用者） |
| getUserInfo | 獲取某使用者的資訊 |
| updateMyInfo | 更新我的資訊（當前登錄使用者） |
| updateMyAvatar | 更新当前用户头像。 |
| getUserAvatar | 获取指定用户的头像缩略图。 |
| getOriginalUserAvatar | 获取指定用户的头像原图，如果在上传用户头像时没有做约束，调用该方法可能会导致 OOM。 |
| sendGroupTextMessage | 發送群聊文本消息 |
| sendGroupImageMessage | 發送群聊圖片消息 |
| \*sendGroupFileMessage | 发送一条群聊文件消息。 |
| \*sendGroupLocationMessage | 发送一条群聊地址消息（可同时用于 iOS）。 |
| getOfflineMessages | 获取上次离线期间收到的离线消息。 |
| getLatestMessage | 获取指定会话中最近的消息。 |
| getHistoryMessages | 获取指定会话中从新到旧的部分历史消息。 |
| getAllMessages | 获取指定会话中的所有消息。 |
| getOriginImageInGroupConversation | 获取指定群聊会话中的指定图片消息原图，第一次调用时会从网络下载原图到本地。 |
| \*createGroup | 創建群組 (Android) |
| getGroupIDList | 得到当前用户加入的所有 Group ID。 |
| getGroupInfo | 获取群组信息。 |
| updateGroupName | 更新群组名称。 |
| updateGroupDescription | 更新群组说明。 |
| updateGroupDescription | 更新群组说明。 |
| \*addGroupMembers | 向群组中添加用户。 |
| \*removeGroupMembers | 从群组中删除指定用户。 |
| exitGroup | 退出群組（指定群組） |
| getGroupMembers | 获取指定群组中的所有用户信息。 |
| getFriendList | 獲取好友列表 |
| sendInvitationRequest | 發送添加好友請求 |
| acceptInvitation | 接受好友邀請 |
| declineInvitation | 拒绝好友请求。 |
| removeFromFriendList | 删除好友，调用后对方会触发 onContactDeleted 事件。 |
| jmessage.onInviteReceived | 收到好友邀请时触发。 |
| isGroupConversationExist | 判断指定群聊会话是否存在。 |
| getConversationList | 获取当前用户的所有会话列表。从本地数据库获得，同步返回。 |
| exitConversation | 退出当前会话。在退出会话界面时需要调用该函数，与 enterSingleConversation / enterGroupConversation 函数配套使用。 |
| enterGroupConversation | 进入特定群聊会话，调用后收到指定会话消息时不会再弹出通知。 |
| getGroupConversation | 获取和特定群组的群聊会话。 |
| getAllGroupConversation | 获取当前用户所有的群聊会话。 |
| setGroupConversationUnreadMessageCount | 设置指定单聊会话的未读消息数。 |
| ##DocumentEvent | 監聽所有事件 |

## User

### \*API - login

用户登录

#### 接口定义

window.JMessage.login(username, password, successCallback, errorCallback);

### API - logout

当前用户退出登录

#### 接口定义

window.JMessage.logout(successCallback, errorCallback);

### API - getMyInfo

获取我的信息（当前登录用户）

#### 接口定义

window.JMessage.getMyInfo(successCallback, errorCallback);

### API - getUserInfo

获取某用户的信息

#### 接口定义

window.JMessage.getUserInfo(username,successCallback, errorCallback);

### API - getUserInfoArray

获取用户详情（批量接口）

#### 接口定义

window.JMessage.getUserInfoArray(usernameArray, successCallback, errorCallback);

### API - updateMyInfo（与 Android 通用）

获取我的信息（当前登录用户）

#### 接口定义

window.JMessage.updateMyInfo(field, value, successCallback, errorCallback);

#### 参数说明

* userInfoField：需要更新的用户信息字段。包括：
  + nickname：昵称。
  + birthday：生日。
  + signature：个性签名。
  + gender：性别。
  + region：地区。
  + avatar : 头像。
* value：更新的值，当更新头像时，为图片路径。
* successCallback：获取成功的回调函数，无返回值。
* errorCallback：更新失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

## Message

### API - sendGroupTextMessage

发送群聊文本消息

#### 接口定义

window.JMessage.sendGroupTextMessage(groupId, text, successCallback, errorCallback);

### API - sendGroupImageMessage

发送群聊图片消息

#### 接口定义

window.JMessage.sendGroupImageMessage (groupId, imageUrl, successCallback, errorCallback);

## Conversation

### API - getGroupConversationHistoryMessage

获取多条群聊消息（同步接口）。参数同 [获取多条单聊消息](https://github.com/jpush/jmessage-phonegap-plugin/blob/master/doc/iOS_API.md#api---getsingleconversationhistorymessage)

#### 接口定义

window.JMessage.getGroupConversationHistoryMessage(groupId, from, limit, successCallback, errorCallback);

#### 参数说明

* from：开始的位置，null 表示从最初开始
* limit：获取的数量，null 表示不限
* 例：
  + from = nil, limit = nil，表示获取全部。相当于 allMessages
  + from = nil, limit = 100，表示从最新开始取 100 条记录
  + from = 100, limit = nil，表示从最新第 100 条开始，获取余下所有记录

### API - getAllGroupConversation

获取全部群聊会话信息

#### 接口定义

window.JMessage.getAllGroupConversation(successCallback, errorCallback);

### API - deleteGroupConversation

删除群聊会话

#### 接口定义

window.JMessage.deleteGroupConversation(groupId, successCallback, errorCallback);

### API - getAllUnreadCount

获取当前所有会话的未读消息的总数

#### 接口定义

window.JMessage.getAllUnreadCount(successCallback, errorCallback);

### API - clearGroupUnreadCount

清除群聊会话未读数

#### 接口定义

window.JMessage.clearGroupUnreadCount(groupId, successCallback, errorCallback);

### \*API - sendFileMessage

发送单聊、群聊文件消息

#### 接口定义

window.JMessage.sendFileMessage(name, appKey, single, filePath, fileName, successCallback, errorCallback);

#### 参数说明

* name：用户名或 groupId
* appkey：对方的 appkey
* single：string，单聊 "1"，群聊 "0"
* filePath：文件路径，/ 开头，不要添加 file: 等额外字符
* fileName：文件名

### \*API - sendLocationMessage

发送地理位置消息

#### 接口定义

window.JMessage.sendLocationMessage(name, appKey, single, latitude, longitude, scale, address, successCallback, errorCallback);

#### 参数说明

* name：用户名或 groupId
* appkey：对方的 appkey
* single：string，单聊 "1"，群聊 "0"
* latitude：string，纬度
* longitude：string，经度
* scale：缩放比例
* address：详细地址

## Group

### API - createGroupIniOS

创建群组

#### 接口定义

window.JMessage.createGroupIniOS(name, desc, memebersArray, successCallback, errorCallback);

### API - updateGroupInfo

修改群组信息

#### 接口定义

window.JMessage.updateGroupInfo(groupId, name, desc, successCallback, errorCallback);

#### 参数说明

* name：新名称
* desc：新描述

### API - getGroupInfo

获取群组详情（不包含群组成员）

#### 接口定义

window.JMessage.getGroupInfo(groupId, successCallback, errorCallback);

### API - myGroupArray

获取我的群组列表

#### 接口定义

window.JMessage.myGroupArray(successCallback, errorCallback);

### API - memberArray

获取当前群组成员列表

#### 接口定义

window.JMessage.memberArray(groupId, successCallback, errorCallback);

### API - addMembers

添加群组成员

#### 接口定义

window.JMessage.addMembers(groupId, memberArray, successCallback, errorCallback) ;

### API - removeMembers

删除群组成员

#### 接口定义

window.JMessage.removeMembers(groupId, memberArray, successCallback, errorCallback);

### API - exitGroup

退出群组（当前用户）

#### 接口定义

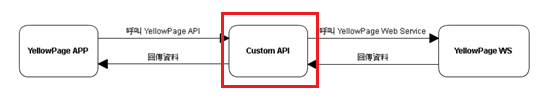
window.JMessage.exitGroup(groupId, successCallback, errorCallback);

## FriendManager

### API - getQFriendList

获取好友列表

### Custom API

****

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appens/getAuthority?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 使用者的員工工號 |
| app\_key | LayoutHeader | Required | string | 必填, 請填入專案所屬的app\_key  ex:  appens  appenstest  appensdev  後台會根據不同的專案獲取不同的Event, Basic info或是權限清單 |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><app\_key>appens</app\_key></LayoutHeader>

因為內文實質上是XML格式, 因此XML保留字元請先做HTMLENCODE

以下5個符號在http post之前需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: FlowER & IP Servers這樣的內文, 需轉換為FlowER &amp; IP Servers後再送出

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **資料名稱** | **母節點** | **是否必填** | **資料類型** | **描述** |
| Role | Content | Y | string | 登入者所屬的角色  目前只有三種  admin:有權限發送緊急通知, 也可以Close Event  supervisor:有權限查看所有通知, 但是沒有Close Event的功能  common:一般USER, 可以收通知, 可以完成子事件 |

\*\*\*\*\*\*\*可能有多重身分, 如下:

{

  "result\_code": 1,

  "content": {

    "RoleList": [

      "admin",

      "supervisor"

    ]

  }

}

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 014908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 014915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 014916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 014917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 014999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

#### 接口定义

window.JMessage.getFriendList(successCallback, errorCallback);

#### 参数说明

* username：对方用户名
* appkey：对方所在应用appkey，不传则默认是本应用
* reason：添加好友时的备注，可不填

### API - sendInvitationRequest

发送添加好友请求

#### 接口定义

window.JMessage.sendInvitationRequest(username, appkey, reason, successCallback, errorCallback);

### API - acceptInvitation

接受好友邀请

#### 接口定义

window.JMessage.acceptInvitation(username, appkey, successCallback, errorCallback);

### API - rejectInvitation

拒绝好友请求

#### 接口定义

window.JMessage.declineInvitation(username, appkey, reason, successCallback, errorCallback);

### API - removeFriend

删除好友

#### 接口定义

window.JMessage.removeFromFriendList(username, appkey, successCallback, errorCallback);

## Disturb

注：用户&群组的免打扰状态、群主 appkey、群最大成员数量在 API - getUserInfo/getGroupInfo 获取。

### \*API - noDisturbList

用户免打扰列表 设置全局免打扰标识

#### 接口定义

window.JMessage.getNoDisturblist(successCallback, errorCallback);

### \*API - isSetGlobalNoDisturb

获取全局免打扰状态

#### 接口定义

window.JMessage.getNoDisturbGlobal(successCallback, errorCallback);

### \*API - setIsGlobalNoDisturb

设置是否全局免打扰

#### 接口定义

window.JMessage.setNoDisturbGlobal(isGlobalNoDisturb, successCallback, errorCallback);

#### 参数说明

isGlobalNoDisturb：1 是/0 否

### \*API - userSetIsNoDisturb

设置用户免打扰（支持跨应用设置）

#### 接口定义

window.JMessage.setUserNoDisturb(username, appkey, isNoDisturb, successCallback, errorCallback);

## DocumentEvent

##### jmessage.onConversationChanged

会话信息变更通知

收到此通知后, 建议处理: 如果 App 当前在会话列表页，刷新整个列表；如果在聊天界面，刷新聊天标题。

##### jmessage.onUnreadChanged

全局未读数变更

##### jmessage.onGroupInfoChanged

群消息变更

##### jmessage.loginUserKicked

登录被踢，已过时，改用 jmessage.onReceiveNotificationEvent

##### jmessage.onReceiveMessage

收到消息

##### jmessage.onSendMessage

发送消息

##### jmessage.onReceiveImageData

收到图片消息

##### \*jmessage.onReceiveVoiceData

收到声音消息

##### \*jmessage.onReceiveFileData

收到文件消息

##### \*jmessage.onReceiveLocation

收到地理信息消息

##### jmessage.onReceiveNotificationEvent

通知事件

// 用户登录状态变更事件

/// 事件类型: 登录被踢

kJMSGEventNotificationLoginKicked = 1,

/// 事件类型: 非客户端修改密码强制登出事件

kJMSGEventNotificationServerAlterPassword = 2,

/// 事件类型：用户登录状态异常事件（需要重新登录）

kJMSGEventNotificationUserLoginStatusUnexpected = 70,

/// 事件类型：当前登录用户信息变更通知事件(非客户端修改)

kJMSGEventNotificationCurrentUserInfoChange = 40,

// 免打扰事件

/// 事件类型: 免打扰变更

kJMSGEventNotificationNoDisturbChange = 37,

/// 事件类型: 黑名单变更

kJMSGEventNotificationBlacklistChange = 38,

// 好友相关事件

/// 事件类型: 好友邀请相关

kJMSGEventNotificationFriendInvitation = 5,

/// 事件类型: 收到好友邀请

kJMSGEventNotificationReceiveFriendInvitation = 51,

/// 事件类型: 对方接受了你的好友邀请

kJMSGEventNotificationAcceptedFriendInvitation = 52,

/// 事件类型: 对方拒绝了你的好友邀请

kJMSGEventNotificationDeclinedFriendInvitation = 53,

/// 事件类型: 对方将你从好友中删除

kJMSGEventNotificationDeletedFriend = 6,

/// 事件类型：非客户端修改好友关系收到好友更新事件

kJMSGEventNotificationReceiveServerFriendUpdate = 7,

// 消息事件

/// 事件类型: 群组被创建

kJMSGEventNotificationCreateGroup = 8,

/// 事件类型: 退出群组

kJMSGEventNotificationExitGroup = 9,

/// 事件类型: 群组添加新成员

kJMSGEventNotificationAddGroupMembers = 10,

/// 事件类型: 群组成员被踢出

kJMSGEventNotificationRemoveGroupMembers = 11,

/// 事件类型: 群信息更新

kJMSGEventNotificationUpdateGroupInfo = 12

# Android Function Description:

Base on Jpush官方所提供的Phonegap API實作QChat API

<https://github.com/jpush/jmessage-phonegap-plugin>

## 集成步骤

* 在线安装
* cordova plugin add jmessage-phonegap-plugin --variable APP\_KEY=your\_app\_key

或

cordova plugin add https://github.com/jpush/jmessage-phonegap-plugin.git --variable APP\_KEY=your\_app\_key

* 本地安装

cordova plugin add <Plugin Path> --variable APP\_KEY=your\_app\_key

## Demo

插件项目中的 /example 目录下包含一个简单的示例，如果想参考可以将目录下的所有文件拷贝到具体 Cordova 项目的 /assets/www/ 目录下。

* Android API名稱列表與描述 (\*表示由QPLAY SERVER端控制)

|  |  |
| --- | --- |
| **Custom API名稱** | **描述** |
| \*login | 用戶登錄 |
| \*logout | 當前用戶退出登錄 |
| getMyInfo | 獲取我的資訊（當前登錄使用者） |
| getUserInfo | 獲取某使用者的資訊 |
| updateMyInfo | 更新我的資訊（當前登錄使用者） |
| updateMyAvatar | 更新当前用户头像。 |
| getUserAvatar | 获取指定用户的头像缩略图。 |
| getOriginalUserAvatar | 获取指定用户的头像原图，如果在上传用户头像时没有做约束，调用该方法可能会导致 OOM。 |
| sendGroupTextMessage | 發送群聊文本消息 |
| sendGroupImageMessage | 發送群聊圖片消息 |
| \*sendGroupFileMessage | 发送一条群聊文件消息。 |
| \*sendGroupLocationMessage | 发送一条群聊地址消息（可同时用于 iOS）。 |
| getOfflineMessages | 获取上次离线期间收到的离线消息。 |
| getLatestMessage | 获取指定会话中最近的消息。 |
| getHistoryMessages | 获取指定会话中从新到旧的部分历史消息。 |
| getAllMessages | 获取指定会话中的所有消息。 |
| getOriginImageInGroupConversation | 获取指定群聊会话中的指定图片消息原图，第一次调用时会从网络下载原图到本地。 |
| \*createGroup | 創建群組 (Android) |
| getGroupIDList | 得到当前用户加入的所有 Group ID。 |
| getGroupInfo | 获取群组信息。 |
| updateGroupName | 更新群组名称。 |
| updateGroupDescription | 更新群组说明。 |
| updateGroupDescription | 更新群组说明。 |
| \*addGroupMembers | 向群组中添加用户。 |
| \*removeGroupMembers | 从群组中删除指定用户。 |
| exitGroup | 退出群組（指定群組） |
| getGroupMembers | 获取指定群组中的所有用户信息。 |
| getFriendList | 獲取好友列表 |
| sendInvitationRequest | 發送添加好友請求 |
| acceptInvitation | 接受好友邀請 |
| declineInvitation | 拒绝好友请求。 |
| removeFromFriendList | 删除好友，调用后对方会触发 onContactDeleted 事件。 |
| jmessage.onInviteReceived | 收到好友邀请时触发。 |
| isGroupConversationExist | 判断指定群聊会话是否存在。 |
| getConversationList | 获取当前用户的所有会话列表。从本地数据库获得，同步返回。 |
| exitConversation | 退出当前会话。在退出会话界面时需要调用该函数，与 enterSingleConversation / enterGroupConversation 函数配套使用。 |
| enterGroupConversation | 进入特定群聊会话，调用后收到指定会话消息时不会再弹出通知。 |
| getGroupConversation | 获取和特定群组的群聊会话。 |
| getAllGroupConversation | 获取当前用户所有的群聊会话。 |
| setGroupConversationUnreadMessageCount | 设置指定单聊会话的未读消息数。 |
| ##DocumentEvent | 監聽所有事件 |

### \*login

用户登录。

#### 接口定义

window.JMessage.login(username, password, successCallback, errorCallback)

#### 参数定义

* username：用户名。
* password：密码。
* successCallback：登录成功的回调函数，无返回信息。
* errorCallback：登录失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

#### 代码示例

window.JMessage.login('username', 'password',

function() {

// 登录成功。

}, function(errorStr) {

console.log(errorStr); // 输出错误信息。

});

### logout

用户登出。

#### 接口定义

window.JMessage.logout(successCallback, errorCallback)

#### 参数定义

* successCallback：登出成功的回调函数，无返回信息。
* errorCallback：登出失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

#### 代码示例

window.JMessage.logout(function() {

// 登出成功。

}, function(errorStr) {

console.log(errorStr); // 输出错误信息。

});

## 用户属性维护

### getUserInfo

获取用户信息。

#### 接口定义

window.JMessage.getUserInfo(username, appKey, successCallback, errorCallback)

#### 参数说明

* username：用户名。
* appKey：目标用户所属应用的 AppKey，可以使用此参数获取不同应用下的用户信息。如果为空，默认获取当前应用下的用户信息。
* successCallback：获取成功的回调函数，返回用户信息对象的 JSON 字符串。
* errorCallback：获取失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

#### 代码示例

window.JMessage.getUserInfo('username', null,

function(response) {

var userInfo = JSON.parse(response);

}, function(errorStr) {

console.log(errorStr); // 输出错误信息。

});

### getMyInfo

获取当前用户的信息。

#### 接口定义

window.JMessage.getMyInfo(successCallback, errorCallback)

#### 参数说明

* successCallback：获取成功的回调函数，返回当前用户信息对象的 JSON 字符串。
* errorCallback：获取失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

#### 代码示例

window.JMessage.getMyInfo(function(response) {

var myInfo = JSON.parse(response);

}, function(errorStr) {

console.log(errorStr); // 输出错误信息。

});

### updateMyInfo

更新当前用户信息。

#### 接口定义

window.JMessage.updateMyInfo(userInfoField, value, successCallback, errorCallback)

#### 参数说明

* userInfoField：需要更新的用户信息字段。包括：
  + nickname：昵称。
  + birthday：生日。
  + signature：个性签名。
  + gender：性别。
  + region：地区。
  + avatar : 头像。
* value：更新的值，当更新头像时，为图片路径。
* successCallback：获取成功的回调函数，无返回值。
* errorCallback：更新失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

#### 代码示例

window.JMessage.updateMyInfo('nickname', 'yourNickname',

function() {

// 更新成功。

}, function(errorMsg) {

console.log(errorMsg); // 输出错误信息。

});

### updateMyAvatar

更新当前用户头像。

#### 接口定义

window.JMessage.updateMyAvatar(avatarFileUrl, successCallback, errorCallback)

#### 参数说明

* avatarFileUrl：头像文件的 URL。
* successCallback：更新成功的回调函数，无返回值。
* errorCallback：更新失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

#### 代码示例

window.JMessage.updateMyAvatar('avatarFileUrl', function() {

// 更新成功。

}, function(errorMsg) {

console.log(errorMsg);

});

### getUserAvatar

获取指定用户的头像缩略图。

#### 接口定义

window.JMessage.getUserAvatar(username, successCallback, errorCallback)

#### 参数说明

* username：指定用户的用户名，如果为空，默认就为当前用户。
* successCallback：以参数形式返回图片路径。
* errorCallback：以参数形式返回错误信息。

#### 代码示例

window.JMessage.getUserAvatar('targetUsername', function (path) {

// Success callback.

}, function (response) {

// Error callback.

})

### getOriginalUserAvatar

获取指定用户的头像原图，如果在上传用户头像时没有做约束，调用该方法可能会导致 OOM。

#### 接口定义

window.JMessage.getOriginalUserAvatar(username, successCallback, errorCallback)

#### 参数说明

* username：指定用户的用户名，如果为空，默认就为当前用户。
* successCallback：以参数形式返回图片路径。
* errorCallback：以参数形式返回错误信息。

#### 代码示例

window.JMessage.getOriginalUserAvatar('targetUsername', function (path) {

// Success callback.

}, function (response) {

// Error callback.

})

## 发送消息

### sendGroupTextMessage

发送一条群聊文本消息。

#### 接口定义

window.JMessage.sendGroupTextMessage(groupId, text, successCallback, errorCallback)

#### 参数说明

* groupId：群组 ID，数值类型。
* text：文本内容。
* successCallback：发送成功的回调函数，以参数形式返回消息对象的 JSON 字符串。
* errorCallback：发送失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

#### 代码示例

window.JMessage.sendSingleTextMessage(5124132141, 'content',

function(response) {

var message = JSON.parse(response);

}, function(errorMsg) {

console.log(errorMsg); // 输出错误信息。

});

### sendGroupImageMessage

发送一条群聊图片消息。

#### 接口定义

window.JMessage.sendGroupTextMessage(groupId, imageUrl, successCallback, errorCallback)

#### 参数说明

* groupId：群组 ID，数值类型。
* imageUrl：图片文件的 URL。
* successCallback：发送成功的回调函数，以参数形式返回消息对象的 JSON 字符串。
* errorCallback：发送失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

#### 代码示例

window.JMessage.sendGroupImageMessage(512412412, 'imageUrl',

function(response) {

var message = JSON.parse(response);

}, function(errorMsg) {

console.log(errorMsg); // 输出错误信息。

});

### \*sendGroupVoiceMessage

发送一条群聊语音消息。

#### 接口定义

window.JMessage.sendGroupVoiceMessage(groupId, voiceFileUrl, successCallback, errorCallback)

#### 参数说明

* groupId：群组 ID，数值类型。
* voiceFileUrl：语音文件的 URL。
* successCallback：发送成功的回调函数，以参数形式返回消息对象的 JSON 字符串。
* errorCallback：发送失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

#### 代码示例

window.JMessage.sendGroupVoiceMessage(151231231, 'voiceFileUrl',

function(response) {

var message = JSON.parse(response);

}, function(errorMsg) {

console.log(errorMsg); // 输出错误信息。

});

### \*sendGroupLocationMessage

发送一条群聊地址消息（可同时用于 iOS）。

#### 接口定义

window.JMessage.sendSingleLocationMessage(groupId, latitude, longtitude, scale, address,

successCallback, errorCallback)

#### 参数说明

* groupId：long，群组 ID。
* latitude：double，纬度信息。
* longtitude：double，经度信息。
* scale：int，地图缩放比例。
* address：String，详细地址。
* successCallback：发送成功的回调函数，以参数形式返回消息对象的 JSON 字符串。
* errorCallback：发送失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

### \*sendGroupFileMessage

发送一条群聊文件消息。

#### 接口定义

window.JMessage.sendSingleFileMessage(groupId, filePath, fileName, successCallback, errorCallback)

#### 参数定义

* groupId：long，群组 ID。
* filePath：String，文件路径。
* fileName：String，文件名称，如果为空，则默认使用文件原名称。
* successCallback：发送成功的回调函数，以参数形式返回消息对象的 JSON 字符串。
* errorCallback：发送失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

## 获取历史消息

### getOfflineMessages

获取上次离线期间收到的离线消息。

#### 接口定义

window.JMessage.getOfflineMessages(successCallback, errorCallback)

#### 参数说明

* successCallback: 成功回调。以参数形式返回离线消息的 JSONArray。
* errorCallback：失败回调。以参数形式返回错误信息。

#### 代码示例

window.JMessage.getOfflineMessages(function (msgArr) {

var content = msgArr[0].content.text // 文本消息内容

}, function (errMsg) {

console.log(errMsg)

})

### getLatestMessage

获取指定会话中最近的消息。

#### 接口定义

window.JMessage.getLatestMessage(conversationType, value, appKey, successCallback, errorCallback)

#### 参数说明

* conversationType：会话类型。有"single", "group"两种。
* value：确定指定会话的参数。如果 conversationType 为 single，即为 username；如果为 group，则为 groupId。
* appKey：当 conversationType 为 single 时，目标用户所属应用的 AppKey。如果为空，默认获得本应用下的会话消息。
* successCallback：发送成功的回调函数，以参数形式返回消息对象的 JSON 字符串。
* errorCallback：发送失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

#### 代码示例

window.JMessage.getLatestMessage('single', 'targetUsername', 'targetAppKey',

function(response) {

var msg = JSON.parse(response);

}, function(errorMsg) {

console.log(errorMsg); // 输出错误信息。

});

window.JMessage.getLatestMessage('group', 'targetGroupId', null,

function(response) {

var msg = JSON.parse(response);

}, function(errorMsg) {

console.log(errorMsg); // 输出错误信息。

});

### getHistoryMessages

获取指定会话中从新到旧的部分历史消息。

#### 接口定义

window.JMessage.getHistoryMessages(conversationType, value, appKey, from, limit, successCallback, errorCallback)

#### 参数说明

* conversationType：会话类型。有"single"，"group"两种。
* value：确定指定会话的参数。当会话类型为 single 时，为 username；会话类型为 group 时，则为 groupId。
* appKey：当 conversationType 为 single 时，目标会话用户所属应用的 AppKey。如果为空，默认获取本应用下的会话消息。
* from：从第几条开始获取历史消息。
* limit：要获取的历史消息数量。
* successCallback：发送成功的回调函数，以参数形式返回消息数组对象的 JSON 字符串。
* errorCallback：发送失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

#### 代码示例

// 获取从最新消息开始的 50 条历史消息。

window.JMessage.getHistoryMessages('single', 'targetUsername', 'targetAppKey', 0, 50,

function(response) {

var messages = JSON.parse(response);

}, function(errorMsg) {

console.log(errorMsg); // 输出错误信息。

});

window.JMessage.getHistoryMessages('group', 'targetGroupId', null, 0, 50,

function(response) {

var messages = JSON.parse(response);

}, function(errorMsg) {

console.log(errorMsg); // 输出错误信息。

});

### getAllMessages

获取指定会话中的所有消息。

#### 接口定义

window.JMessage.getAllMessages(conversationType, value, appKey, successCallback, errorCallback)

#### 参数说明

* conversationType：String，会话类型。有"single"，"group"两种。
* value：String，确定指定会话的参数。当会话类型为 single 时，为 username；会话类型为 group 时，则为 groupId。
* appKey：String，当 conversationType 为 single 时，目标会话用户所属应用的 AppKey。如果为空，默认获取本应用下的会话消息。
* successCallback：发送成功的回调函数，以参数形式返回消息数组对象的 JSON 字符串。
* errorCallback：发送失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

#### 代码示例

window.JMessage.getAllMessages('single', 'targetUsername', 'targetAppKey',

function(response) {

var messages = JSON.parse(response)

}, function(errorMsg) {

console.log(errorMsg) // 输出错误信息。

})

window.JMessage.getAllMessages('group', 'targetGroupId', null,

function(response) {

var messages = JSON.parse(response)

}, function(errorMsg) {

console.log(errorMsg) // 输出错误信息。

})

### getOriginImageInGroupConversation

获取指定群聊会话中的指定图片消息原图，第一次调用时会从网络下载原图到本地。

#### 接口定义

window.JMessage.getOriginImageInGroupConversation(groupId, serverMessageId, successCallback, errorCallback)

#### 参数说明

* groupId：指定群聊会话的 Group ID。
* serverMessageId：图片消息的 serverMessageId。
* successCallback：以参数形式返回图片地址。
* errorCallback：以参数形式返回错误信息。

#### 代码示例

window.JMessage.getOriginImageInGroupConversation(151231241, 83708669, function (path) {

// Success callback.

}, function (errorMsg) {

// Error callback.

})

## 好友关系

### sendInvitationRequest

发送添加好友请求。

#### 接口定义

window.JMessage.sendInvitationReques(targetUsername, targetUserAppkey, reason, successCallback, errorCallback)

#### 参数说明

* targetUsername：String，目标用户的用户名。
* targetUserAppkey：String，目标用户的 AppKey，如果为空则默认为本应用下的用户。
* reason：申请理由。
* successCallback：成功回调。
* errorCallback：错误回调，以参数形式返回错误信息。

### acceptInvitation

接受好友请求。

#### 接口定义

window.JMessage.acceptInvitation(targetUsername, targetUserAppkey, successCallback, errorCallback)

#### 参数说明

* targetUsername：String，目标用户的用户名。
* targetUserAppkey：String，目标用户的 AppKey，如果为空则默认为本应用下的用户。
* successCallback：成功回调。
* errorCallback：错误回调，以参数形式返回错误信息。

### declineInvitation

拒绝好友请求。

#### 接口定义

window.JMessage.declineInvitation(targetUsername, targetUserAppkey, reason, successCallback, errorCallback)

#### 参数说明

* targetUsername：String，目标用户的用户名。
* targetUserAppkey：String，目标用户的 AppKey，如果为空则默认为本应用下的用户。
* reason：String，拒绝理由。
* successCallback：成功回调。
* errorCallback：错误回调，以参数形式返回错误信息。

### getFriendList

获取当前登录用户的好友列表。

#### 接口定义

window.JMessage.getFriendList(successCallback, errorCallback)

#### 参数说明

* successCallback：成功回调，以参数形式返回好友信息数组的 JSON 字符串。
* errorCallback：错误回调，以参数形式返回错误信息。

### removeFromFriendList

删除好友，调用后对方会触发 onContactDeleted 事件。

#### 接口定义

window.JMessage.removeFromFriendList(targetUsername, targetUserAppkey, successCallback, errorCallback)

#### 参数说明

* targetUsername：String，目标用户的用户名。
* targetUserAppkey：String，目标用户的 AppKey，如果为空则默认为本应用下的用户。
* successCallback：成功回调。
* errorCallback：错误回调，以参数形式返回错误信息。

### jmessage.onInviteReceived

收到好友邀请时触发。

#### 参数说明

回调函数的参数为一个 JSON 数据字符串。其格式为：

{

reason：'', // 事件发生的理由，该字段由对方发起请求时所填，对方如果未填则将返回默认字符串

fromUsername: '', // 事件发送者的 username

fromAppKey: '' // 事件发送者的 AppKey

}

## 聊天会话

### isGroupConversationExist

判断指定群聊会话是否存在。

#### 接口定义

window.JMessage.isGroupConversationExist(groupId, successCallback, errorCallback)

#### 参数说明

* groupId：群组 ID，数值类型。
* successCallback：以参数形式返回结果。0: 不存在；1：存在。
* errorCallback：以参数形式返回错误信息。

### getConversationList

获取当前用户的所有会话列表。从本地数据库获得，同步返回。

#### 接口定义

window.JMessage.getConversationList(successCallback, errorCallback)

#### 参数说明

* successCallback：发送成功的回调函数，以参数形式返回会话数组对象的 JSON 字符串。
* [
* {
* "id": "56740fc3-25e0-468d-a490-d644470d63d2", // Conversation ID
* "latestType": "最近一条消息的类型",
* "latestText": "最近一条消息的内容",
* "targetId": "目标用户的用户名",
* "title": "会话标题",
* "type": "会话类型（single / group）",
* "unReadMsgCnt": 0, // 未读消息数
* "lastMsgDate": 1468983461848 // 最近消息的收到时间，单位为 ms
* }
* ]
* errorCallback：发送失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

#### 代码示例

window.JMessage.getConversationList(function(response) {

var conversations = JSON.parse(response);

}, function(errorMsg) {

console.log(errorMsg); // 输出错误信息。

});

### exitConversation

退出当前会话。在退出会话界面时需要调用该函数，与 enterSingleConversation / enterGroupConversation 函数配套使用。

#### 接口定义

window.JMessage.exitConversation(successCallback, errorCallback)

#### 参数说明

* successCallback：发送成功的回调函数，无返回值。
* errorCallback：发送失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

#### 代码示例

window.JMessage.exitConversation(function() {

// 退出成功。

}, function(errorMsg) {

console.log(errorMsg); // 输出错误信息。

});

### 群聊

#### enterGroupConversation

进入特定群聊会话，调用后收到指定会话消息时不会再弹出通知。

##### 接口定义

window.JMessage.enterGroupConversation(groupId, successCallback, errorCallback)

##### 参数说明

* groupId：目标群组 ID。
* successCallback：获取成功的回调函数，无返回值。
* errorCallback：获取失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

##### 代码示例

window.JMessage.enterGroupConversation('targetGroupId',

function() {

// 进入会话成功。

}, function(errorMsg) {

console.log(errorMsg)

})

#### getGroupConversation

获取和特定群组的群聊会话。

##### 接口定义

window.JMessage.getGroupConversation(groupId, successCallback, errorCallback)

##### 参数说明

* groupId：群组 ID。
* successCallback：发送成功的回调函数，以参数形式返回会话对象的 JSON 字符串。
* errorCallback：发送失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

##### 代码示例

window.JMessage.getGroupConversation('targetGroupId',

function(response) {

var groupConversation = JSON.parse(reponse);

}, function(errorMsg) {

console.log(errorMsg); // 输出错误信息。

});

#### getAllGroupConversation

获取当前用户所有的群聊会话。

##### 接口定义

window.JMessage.getAllGroupConversation(successCallback, errorCallback)

##### 参数说明

* successCallback：发送成功的回调函数，以参数形式返回会话列表对象的 JSON 字符串。
* errorCallback：发送失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

##### 代码示例

window.JMessage.getAllGroupConversation(function(response) {

var groupConversations = JSON.parse(response);

}, function(errorMsg) {

console.log(errorMsg);

});

#### deleteGroupConversation

删除指定的群聊会话。

##### 接口定义

window.JMessage.deleteGroupConversation(groupId, successCallback, errorCallback)

##### 参数说明

* groupId：群组 ID。
* successCallback：删除成功的回调函数，无返回值。
* errorCallback：删除失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

##### 代码示例

window.JMessage.deleteGroupConversation('targetGroupId',

function() {

}, function(errorMsg) {

console.log(errorMsg);

});

#### setGroupConversationUnreadMessageCount

设置指定单聊会话的未读消息数。

##### 接口定义

window.JMessage.setGroupConversationUnreadMessageCount(groupId, unreadCount, successCallback, errorCallback)

##### 参数说明

* groupId：群组 ID。
* unreadCount：未读消息数。
* successCallback：设置成功的回调函数，无返回值。
* errorCallback：设置失败的回调函数，以参数形式返回错误信息。如果为 null，默认打印失败信息日志。

##### 代码示例

window.JMessage.setGroupConversationUnreadMessageCount('targetGroupId', 10,

function() {

// 设置成功。

}, function(errorMsg) {

console.log(errorMsg);

});

## 群组

### createGroup

创建群组。

#### 接口定义

window.JMessage.createGroup(groupName, groupDesc, successCallback, errorCallback)

#### 参数说明

* groupName：群组名。
* groupDesc：群组描述。
* successCallback：以参数形式返回 group ID。
* errorCallback：以参数形式返回错误信息。

#### 代码示例

window.JMessage.createGroup('groupName', 'groupDesc', function (groupId) {

// Success callback.

}, function (errorMsg) {

// Error callback.

})

### getGroupIDList

得到当前用户加入的所有 Group ID。

#### 接口定义

window.JMessage.getGroupIDList(successCallback, errorCallback)

#### 参数说明

* successCallback：以参数形式返回所有 Group ID 的 JSON 字符串。
* errorCallback：以参数形式返回错误信息。

#### 代码示例

window.JMessage.getGroupIDList(function (json) {

// Success callback.

}, function (errorMsg) {

// Error callback.

})

### getGroupInfo

获取群组信息。

#### 接口定义

window.JMessage.getGroupInfo(groupId, successCallback, errorCallback)

#### 参数说明

* groupId：目标群组的 ID。
* successCallback：以参数形式返回群组信息的 JSON 字符串。
* errorCallback：以参数形式返回错误信息。

#### 代码示例

window.JMessage.getGroupInfo(1512314121, function (json) {

// Success callback.

}, function (errorMsg) {

// Error callback.

})

### updateGroupName

更新群组名称。

#### 接口定义

window.JMessage.updateGroupName(groupId, groupNewName, successCallback, errorCallback)

#### 参数说明

* groupId：long，群组 ID。
* groupNewName：String，群组新名称。
* successCallback：无返回值。
* errorCallback：以参数形式返回错误信息。

#### 代码示例

window.JMessage.updateGroupName(14123123, 'newName', function () {

// Success callback.

}, function (errorMsg) {

// Error callback.

})

### \*updateGroupDescription

更新群组说明。

#### 接口定义

window.JMessage.updateGroupDescription(groupId, groupNewDesc, successCallback, errorCallback)

#### 参数说明

* groupId：群组 ID。
* groupNewDesc：群组描述。
* successCallback：无返回值。
* errorCallback：以参数形式返回错误码。

#### 代码示例

window.JMessage.updateGroupDescription(115123121, 'newDesc', function () {

// Success callback.

}, function (errorMsg) {

// Error callback.

})

### addGroupMembers

向群组中添加用户。

#### 接口定义

window.JMessage.addGroupMembers(groupId, usernameStr, successCallback, errorCallback)

#### 参数说明

* groupId：群组 ID。
* usernameStr：要添加的用户名字符串，形如：'username1,username2'。
* successCallback：无返回值。
* errorCallback：以参数形式返回错误码。

#### 代码示例

window.JMessage.addGroupMembers(15131231, 'username1,username2', function () {

// Success callback.

}, function (errorMsg) {

// Error callback.

})

### removeGroupMembers

从群组中删除指定用户。

#### 接口定义

window.JMessage.removeGroupMembers(groupId, usernameStr, successCallback, errorCallback)

#### 参数说明

* groupId：群组 ID。
* usernameStr：要删除的用户名字符串，形如：'username1,username2'。
* successCallback：无返回值。
* errorCallback：以参数形式返回错误码。

#### 代码示例

window.JMessage.addGroupMembers(15131231, 'username1,username2', function () {

// Success callback.

}, function (errorMsg) {

// Error callback.

})

### exitGroup

退出指定群组。

#### 接口定义

window.JMessage.exitGroup(groupId, successCallback, errorCallback)

#### 参数说明

* groupId：群组 ID。
* successCallback：无返回值。
* errorCallback：以参数形式返回错误码。

#### 代码示例

window.JMessage.addGroupMembers(15131231, function () {

// Success callback.

}, function (errorMsg) {

// Error callback.

})

### getGroupMembers

获取指定群组中的所有用户信息。

#### 接口定义

window.JMessage.getGroupMembers(groupId, successCallback, errorCallback)

#### 参数说明

* groupId：群组 ID。
* successCallback：以参数形式返回用户信息的 JSON 字符串。
* errorCallback：以参数形式返回错误码。

#### 代码示例

window.JMessage.getGroupMembers(15131231, function (json) {

// Success callback.

}, function (errorMsg) {

// Error callback.

})

## 事件处理

### 消息事件

#### 消息对象的 JSON 数据格式

{

"content": {

"text": "content",

"booleanExtras": { },

"contentType": "text",

"extras": {

"a": { }

},

"numExtras": { },

"stringExtras": { // 附加字段。

"key1": "value1",

"key2": "value2"

}

},

"contentType": "text",

"createTimeInSeconds": 1466496463,

"direct": "receive",

"fromAppkey": "fromUserAppKey",

"fromType": "user",

"fromUser": {

"address": "",

"appkey": "yourAppKey",

"birthday": "",

"gender": "0",

"mGender": "male",

"nickname": "targetNickname",

"noteText": "",

"notename": "",

"region": "",

"signature": "",

"userName": "testUsername",

"userID": testUserID,

"blacklist": 0,

"noDisturb": 0,

"star": 0,

"avatarPath": "/data/user/0/io.cordova.hellocordova/files/images/small-avatar/avatarName" // 发送用户的头像缩略图。

},

"from\_platform": "a",

"msgTypeString": "text",

"serverMessageId": 73511240,

"status": "receive\_success",

"targetAppkey": "targetAppkey",

"targetInfo": {

"address": "",

"appkey": "targetAppKey",

"birthday": "",

"gender": "0",

"mGender": "female",

"nickname": "testNickname",

"noteText": "",

"notename": "",

"region": "",

"signature": "",

"userName": "testUsername",

"userID": testUserID,

"blacklist": 0,

"noDisturb": 0,

"star": 0,

"avatarPath": "/data/user/0/io.cordova.hellocordova/files/images/small-avatar/avatarName" // 接收用户的头像缩略图。

},

"targetName": "",

"targetType": "single",

"version": 1,

"\_id": 7,

"createTimeInMillis": 1466496463000

}

#### jmessage.onSyncOfflineMessage（同时适用于 iOS）

当在上次离线期间收到消息时触发。

##### 代码示例

document.addEventListener('jmessage.onSyncOfflineMessage', function (evnet) {

var conversation = event.conversation // 会话对象

var messageArr = event.messageList // 离线消息数组

}, false)

#### jmessage.onOpenMessage

点击通知栏中的消息通知时触发。

##### 代码示例

document.addEventListener('jmessage.onOpenMessage', function(msg) {

}, false);

#### jmessage.onReceiveMessage

收到消息时触发。

##### 代码示例

document.addEventListener('jmessage.onReceiveMessage', function(msg) {

}, false);

#### jmessage.onReceiveTextMessage

收到文本消息触发。

##### 代码示例

document.addEventListener('jmessage.onReceiveTextMessage', function(msg) {

}, false);

#### jmessage.onReceiveImageMessage

收到图片消息触发。

##### 代码示例

document.addEventListener('jmessage.onReceiveImageMessage', function(msg) {

}, false);

#### jmessage.onReceiveVoiceMessage

收到语音消息触发。

##### 代码示例

document.addEventListener('jmessage.onReceiveVoicetMessage', function(msg) {

}, false);

#### jmessage.onReceiveCustomMessage

收到自定义消息触发。

##### 代码示例

document.addEventListener('jmessage.onReceiveCustomMessage', function(msg) {

}, false);

### 用户状态变更事件

#### jmessage.onUserPasswordChanged

当用户密码在服务器端被修改时触发。

##### 代码示例

document.addEventListener('jmessage.onUserPasswordChanged', yourFunction, false);

#### jmessage.onUserLogout

当用户换设备登录时触发。

##### 代码示例

document.addEventListener('jmessage.onUserLogout', yourFunction, false);

#### jmessage.onUserDeleted

当用户被删除时触发。

##### 代码示例

document.addEventListener('jmessage.onUserDeleted', yourFunction, false);

### 群组事件

#### jmessage.onGroupMemberAdded

群成员加群时触发。

##### 代码示例

document.addEventListener('jmessage.onGroupMemberAdded', yourFunction, false);

#### jmessage.onGroupMemberRemoved

群成员被踢时触发。

##### 代码示例

document.addEventListener('jmessage.onGroupMemberRemoved', yourFunction, false);

#### jmessage.onGroupMemberExit

群成员退群时触发。

##### 代码示例

document.addEventListener('jmessage.onGroupMemberExit', yourFunction, false);

### 好友事件

#### jmessage.onInviteReceived

当收到添加好友申请。

#### jmessage.onInviteAccepted

当前用户发送的好友请求被接受。

#### jmessage.onInviteDeclined

当前用户发送的好友请求被拒绝。

#### jmessage.onContactDeleted

当对方将你从好友列表中删除。

# QChat Custom API

## 好友管理

### getQList

#### 接口定義

透過此API可以獲得集團內的人員基本資料

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/getQList?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 使用者的員工工號 |
| search\_type | LayoutHeader | Required | string | 1:name  2:department  3:only same department  只找尋同公司同部門的人 |
| search\_string | LayoutHeader | Required | string | 當search\_type = 1 or 2時, search\_string欄位必填  當search\_type=3時, search\_string是非必填  會直接按emp\_no查找相同公司,相同部門的人  必填, 請填入要搜尋的字串  ex:  steven  steven.yan  yan  也可以填入部門碼:  ex: BI  後台會根據模糊查詢返回結果 |
| friend\_only | LayoutHeader | Required | string | 必填  Y:只找朋友  N:全找 |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><search\_string>steven</search\_string><friend\_only>N</friend\_only></LayoutHeader>

因為post body內文實質上是XML格式, 因此XML某些保留字元需要先做HTMLENCODE

以下5個符號在http post之前, 各節點的value如果有含以下左邊原始保留字元, 需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: <event\_title>FlowER & IP Servers</event\_title>這樣的內文, 需轉換為<event\_title>FlowER &amp; IP Servers</event\_title>後再送出, &符號需要處理

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **資料名稱** | **母節點** | **出現次數** | **資料類型** | **描述** |
| user\_list | Content | 0-1 | Container |  |
| name | user\_list | 0-N | string | 回傳AD帳號  ex:Steven.Yan  字首皆是大寫 |
| registered | user\_list | 0-N | string | 目前有兩種狀態, 表達是否已經註冊過qplay,  N表示沒有註冊過qplay, 可以寄送邀請函請對方安裝qplay  Y:registered user  N:non-register user |
| status | user\_list | 0-N | string | 目前有三種狀態:  0:無狀態  1:已是好友  2:送出交友邀請中  3:拒絕邀請 |
| protected | user\_list | 0-N | string | 目前有兩種狀態:  Y:protect user  N:normal user |
| emp\_no | user\_list | 0-N | String | 員工工號 |
| domain | user\_list | 0-N | String | 員工所屬domain  ex:Qgroup, BenQ |
| site\_code | user\_list | 0-N | String | 員工所屬地區  ex:QTY,QTT,QCS |
| email | user\_list | 0-N | String | Email |
| ext\_no | user\_list | 0-N | String | 分機 |
| memo | user\_list | 0-N | String | User的狀態消息 |
| portrait\_path | user\_list | 0-N | String | 大頭照的縮圖路徑URL |
| over\_threshold | Content | 0-1 | String | Y=搜尋結果超過預計筆數  N=搜尋結果不超過預計筆數 |

Check point:

* + - 1. 回傳結果, 只抓qp\_user.status=Y
      2. Registered=Y的判斷條件, 須以register\_message為主

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025998 | 查無資料 | No Data | 查无資料 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_friend\_matrix**

**qp\_protect\_user**

**qp\_user**

### getQFriend

#### 接口定義

透過此API可以獲得登入者的所有朋友名單, 包含保護名單已經成為朋友者

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/getQFriend?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 使用者的員工工號 |
| search\_string | LayoutHeader | Optional | string | 選填  如果沒放就回傳所有朋友  請填入要搜尋的字串  ex:  steven  steven.yan  yan  也可以填入部門碼:  ex: BI  後台會根據模糊查詢返回結果 |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><search\_string>steven</search\_string></LayoutHeader>

因為post body內文實質上是XML格式, 因此XML某些保留字元需要先做HTMLENCODE

以下5個符號在http post之前, 各節點的value如果有含以下左邊原始保留字元, 需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: <event\_title>FlowER & IP Servers</event\_title>這樣的內文, 需轉換為<event\_title>FlowER &amp; IP Servers</event\_title>後再送出, &符號需要處理

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **資料名稱** | **母節點** | **出現次數** | **資料類型** | **描述** |
| friend | Content | 1 | Container | 好友區塊 |
| user\_list | friend | 0-1 | Container | 好友列表 |
| name | friend.user\_list | 0-N | string | 回傳AD帳號  ex:Steven.Yan  字首皆是大寫 |
| registered | friend.user\_list | 0-N | string | 目前有兩種狀態:  Y:registered user  N:non-register user |
| status | friend.user\_list | 0-N | string | 目前有三種狀態:  0:無狀態  1:已是好友  2:送出交友邀請中  3:拒絕邀請 |
| protected | friend.user\_list | 0-N | string | 目前有兩種狀態:  Y:protect user  N:normal user |
| emp\_no | friend.user\_list | 0-N | String | 員工工號 |
| domain | friend.user\_list | 0-N | String | 員工所屬domain  ex:Qgroup, BenQ |
| site\_code | friend.user\_list | 0-N | String | 員工所屬地區  ex:QTY,QTT,QCS |
| email | friend.user\_list | 0-N | String | Email |
| ext\_no | friend.user\_list | 0-N | String | 分機 |
| memo | friend.user\_list | 0-N | String | User的狀態消息 |
| portrait\_path | friend.user\_list | 0-N | String | 大頭照的縮圖路徑URL |
| over\_threshold | friend | 0-1 | String | Y=搜尋結果超過預計筆數  N=搜尋結果不超過預計筆數 |
| inviter | Content | 1 | Container | 邀請加入好友區塊 |
| user\_list | Inviter |  |  |  |
| name | inviter.user\_list | 0-N | string | 回傳AD帳號  ex:Steven.Yan  字首皆是大寫 |
| emp\_no | inviter.user\_list | 0-N | String | 員工工號 |

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025998 | 查無資料 | No Data | 查无資料 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_friend\_matrix**

**qp\_protect\_user**

**qp\_user**

### setQFriend

#### 接口定義

透過API設定為好友, 比較貼近設定為我的最愛的概念, 只能針對非保護名單設定好友

保護名單需要使用sendQInvitation這支API送出邀請

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/setQFriend?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 使用者的員工工號 |
| destination\_emp\_no | LayoutHeader | Required | string | 必填, 請填入完整的工號  會從後台比對, 如果不符合則會reject |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><destination\_emp\_no>1607279</destination\_emp\_no></LayoutHeader>

因為post body內文實質上是XML格式, 因此XML某些保留字元需要先做HTMLENCODE

以下5個符號在http post之前, 各節點的value如果有含以下左邊原始保留字元, 需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: <event\_title>FlowER & IP Servers</event\_title>這樣的內文, 需轉換為<event\_title>FlowER &amp; IP Servers</event\_title>後再送出, &符號需要處理

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：NA**

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025921 | 要設定的好友工號不存在 | The destination employee number is invalid | 要设定的好友工号不存在 |
| 025922 | 要設定的好友是保護名單 | The destination employee number is protect user | 要设定的好友是保护名单 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_friend\_matrix**

**qp\_protect\_user**

**qp\_user**

### sendQInvitation

#### 接口定義

透過此API送出交友邀請

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/sendQInvitation?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 使用者的員工工號 |
| destination\_emp\_no | LayoutHeader | Required | string | 必填, 請填入完整的工號  會從後台比對, 如果不符合則會reject |
| reason | LayoutHeader | Optional | string | 請求理由 |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><destination\_emp\_no>1607279</destination\_emp\_no><reason>QPlay Project</reason></LayoutHeader>

因為post body內文實質上是XML格式, 因此XML某些保留字元需要先做HTMLENCODE

以下5個符號在http post之前, 各節點的value如果有含以下左邊原始保留字元, 需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: <event\_title>FlowER & IP Servers</event\_title>這樣的內文, 需轉換為<event\_title>FlowER &amp; IP Servers</event\_title>後再送出, &符號需要處理

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：NA**

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025921 | 要設定的好友工號不存在 | The destination employee number is invalid | 要设定的好友工号不存在 |
| 025923 | 要設定對象已經是好友 | You cannot send the invitation to friends. | 要设定对象已经是好友 |
| 025927 | 已發送過交友邀請 | Invitation already send | 已发送过交友邀请 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_friend\_matrix**

**qp\_protect\_user**

**qp\_user**

### sendQInstall

#### 接口定義

透過此API送出安裝邀請

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/sendQInstall?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 使用者的員工工號 |
| destination\_emp\_no | LayoutHeader | Required | string | 必填, 請填入完整的工號  會從後台比對, 如果不符合則會reject |
| reason | LayoutHeader | Optional | string | 請求理由 |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><destination\_emp\_no>1607279</destination\_emp\_no><reason>Discussion requirement</reason></LayoutHeader>

因為post body內文實質上是XML格式, 因此XML某些保留字元需要先做HTMLENCODE

以下5個符號在http post之前, 各節點的value如果有含以下左邊原始保留字元, 需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: <event\_title>FlowER & IP Servers</event\_title>這樣的內文, 需轉換為<event\_title>FlowER &amp; IP Servers</event\_title>後再送出, &符號需要處理

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：NA**

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025921 | 要設定的好友工號不存在 | The destination employee number is invalid | 要设定的好友工号不存在 |
| 025924 | 要邀請的好友已經註冊過QPlay | The destination employee already registered | 要邀请的好友已经注册过QPlay |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_friend\_matrix**

**qp\_protect\_user**

**qp\_user**

### getQUserDetail

#### 接口定義

透過此API可以獲得人員的詳細資訊, 包含分機號碼等

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/getQUserDetail?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 使用者的員工工號 |
| destination\_emp\_no | LayoutHeader | Required | string | 必填, 請填入完整的工號  會從後台比對, 如果不符合則會reject |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><destination\_emp\_no>0407731</destination\_emp\_no></LayoutHeader>

因為post body內文實質上是XML格式, 因此XML某些保留字元需要先做HTMLENCODE

以下5個符號在http post之前, 各節點的value如果有含以下左邊原始保留字元, 需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: <event\_title>FlowER & IP Servers</event\_title>這樣的內文, 需轉換為<event\_title>FlowER &amp; IP Servers</event\_title>後再送出, &符號需要處理

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **資料名稱** | **母節點** | **出現次數** | **資料類型** | **描述** |
| name | Content | 0-1 | string | 回傳AD帳號  ex:Steven.Yan  字首皆是大寫 |
| registered | Content | 0-1 | string | 目前有兩種狀態:  Y:registered user  N:non-register user |
| status | Content | 0-1 | string | 目前有三種狀態:  0:無狀態  1:已是好友  2:送出交友邀請中  3:拒絕邀請 |
| protected | Content | 0-1 | string | 目前有兩種狀態:  Y:protect user  N:normal user |
| ~~loginid~~ | ~~content~~ | ~~0-1~~ | ~~String~~ | ~~員工AD帳號~~  ~~ex:steven.yan~~ |
| emp\_no | content | 0-1 | String | 員工工號 |
| domain | content | 0-1 | String | 員工所屬domain  ex:Qgroup, BenQ |
| site\_code | content | 0-1 | String | 員工所屬地區  ex:QTY,QTT,QCS |
| email | content | 0-1 | String | Email |
| ext\_no | content | 0-1 | String | 分機 |
| memo | content | 0-1 | String | User的狀態消息 |
| portrait\_path | content | 0-1 | String | 大頭照的縮圖路徑URL  ex:  https://qplay.benq.com/qplayApi/public/portrait/original/0407731/0407731\_portrait.jpg |

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025998 | 查無資料 | No Data | 查无資料 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_qchat\_user\_detail**

### setQUserDetail

#### 接口定義

透過此API可以設定人員的memo

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

<https://qplay.benq.com/qplayApi/public/v101/custom/appim/setQUserDetail?lang=en-us&uuid=xxxxx>

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 使用者的員工工號 |
| destination\_emp\_no | LayoutHeader | Required | string | 必填, 請填入被修改人員的完整工號  會從後台比對, 如果不符合則會reject |
| memo | LayoutHeader | Optional | string | User的狀態消息  長度限制200個字 |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><destination\_emp\_no>0407731</destination\_emp\_no><memo>下周請假</memo></LayoutHeader>

因為post body內文實質上是XML格式, 因此XML某些保留字元需要先做HTMLENCODE

以下5個符號在http post之前, 各節點的value如果有含以下左邊原始保留字元, 需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: <event\_title>FlowER & IP Servers</event\_title>這樣的內文, 需轉換為<event\_title>FlowER &amp; IP Servers</event\_title>後再送出, &符號需要處理

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：NA**

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025925 | 不得修改他人資料 | Cannot modify other people’s basic information | 不得修改他人数据 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**若emp\_no及destination\_emp\_no不一致, 則回拋025925**

**影響DB**

**qp\_qchat\_user\_detail**

### ~~setQUserAvator~~ (不使用)

#### 接口定義

透過此API可以設定人員的大頭照

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

<https://qplay.benq.com/qplayApi/public/v101/custom/appim/setQUserAvator?lang=en-us&uuid=xxxxx>

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | **multipart/form-data** |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| portrait\_data | NA | Required | string | 大頭照 |

範例:

<89504e47 0d0a1a0a 0000000d 49484452 000000b4 000000b4 08020000 00b2af91 65000020 00494441 5478012c dd79b724 6b7616f6 8c888c88 8c9c8733 55ddb1d5 6a0db486 06218401 ......

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：NA**

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_qchat\_user\_detail**

### removeQFriend

#### 接口定義

透過此API刪除好友

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/removeQFriend?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 使用者的員工工號 |
| destination\_emp\_no | LayoutHeader | Required | string | 必填, 請填入完整的工號  會從後台比對, 如果不符合則會reject |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><destination\_emp\_no>1607279</destination\_emp\_no></LayoutHeader>

因為post body內文實質上是XML格式, 因此XML某些保留字元需要先做HTMLENCODE

以下5個符號在http post之前, 各節點的value如果有含以下左邊原始保留字元, 需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: <event\_title>FlowER & IP Servers</event\_title>這樣的內文, 需轉換為<event\_title>FlowER &amp; IP Servers</event\_title>後再送出, &符號需要處理

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：NA**

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025921 | 要設定的好友工號不存在 | The destination employee number is invalid | 要设定的好友工号不存在 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_friend\_matrix**

**qp\_protect\_user**

**qp\_user**

#### Check Point

* + - * 1. 一般名般刪除好友，為單向刪除 A(一般名單) 刪B，僅刪除A端好友列表
        2. 保護名單刪除好友，為雙向刪除 A(保護名單)刪B，刪除 AB間所有好友關係

### acceptQInvitation

#### 接口定義

接受交友邀請

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/acceptQInvitation?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 使用者的員工工號 |
| source\_emp\_no | LayoutHeader | Required | string | 必填, 請填入欲接受的發送者工號  會從後台比對, 如果不符合則會reject |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><source\_emp\_no>1607279</source\_emp\_no></LayoutHeader>

因為post body內文實質上是XML格式, 因此XML某些保留字元需要先做HTMLENCODE

以下5個符號在http post之前, 各節點的value如果有含以下左邊原始保留字元, 需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: <event\_title>FlowER & IP Servers</event\_title>這樣的內文, 需轉換為<event\_title>FlowER &amp; IP Servers</event\_title>後再送出, &符號需要處理

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：NA**

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025921 | 要設定的好友工號不存在 | The destination employee number is invalid | 要设定的好友工号不存在 |
| 025928 | 交友邀請不存在 | Invitation not exist | 交友邀请不存在 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_friend\_matrix**

**qp\_protect\_user**

**qp\_user**

#### CheckPoint

1.接受交友邀請後，會建立雙方好友關係，在qp\_friend\_matrix會有兩筆紀錄

### rejectQInvitation

#### 接口定義

拒絕交友邀請

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/rejectQInvitation?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 使用者的員工工號 |
| source\_emp\_no | LayoutHeader | Required | string | 必填, 請填入欲拒絕的發送者工號  會從後台比對, 如果不符合則會reject |
| reason | LayoutHeader | Optional | string | 拒絕的理由 |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><source\_emp\_no>1607279</source\_emp\_no></LayoutHeader>

因為post body內文實質上是XML格式, 因此XML某些保留字元需要先做HTMLENCODE

以下5個符號在http post之前, 各節點的value如果有含以下左邊原始保留字元, 需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: <event\_title>FlowER & IP Servers</event\_title>這樣的內文, 需轉換為<event\_title>FlowER &amp; IP Servers</event\_title>後再送出, &符號需要處理

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：NA**

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025921 | 要設定的好友工號不存在 | The destination employee number is invalid | 要设定的好友工号不存在 |
| 025928 | 交友邀請不存在 | Invitation not exist | 交友邀请不存在 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_friend\_matrix**

**qp\_protect\_user**

**qp\_user**

## 聊天室管理

### newQChatroom

#### 接口定義

透過此API可以新增聊天室

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/newQChatroom?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 手機持有人的員工工號 |
| ~~lang~~ | ~~LayoutHeader~~ | ~~Required~~ | ~~string~~ | ~~ex:en-us~~ |
| ~~need\_push~~ | ~~LayoutHeader~~ | ~~Required~~ | ~~string~~ | ~~是否需要推播~~  ~~ex:Y, N~~ |
| ~~app\_key~~ | ~~LayoutHeader~~ | ~~Required~~ | ~~string~~ | ~~ex:appim~~ |
| chatroom\_name | LayoutHeader | Required | string | 聊天室標題  長度限制64byte(JMessage聊天室標題最大長度限制) |
| chatroom\_desc | LayoutHeader | Required | string | 聊天室描述, 用來存放給local讀取的資訊  ex:  是否需要歷史訊息  群聊或是私聊  定義如下:  need\_history=Y;group\_message=Y |
| member\_list | LayoutHeader | Required | Container |  |
| destination\_emp\_no | member\_list | Required | string | 欲新增的聊天室成員  可以有多個  <destination\_emp\_no>0407732</destination\_emp\_no>  <destination\_emp\_no>0407733</destination\_emp\_no>  <destination\_emp\_no>0407734</destination\_emp\_no> |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><lang>en-us</lang><need\_push>Y</need\_push>

<app\_key>appim</app\_key>

<chatroom\_name>這是一個聊天室</chatroom\_title>

<chatroom\_desc>need\_history=Y;group\_message=Y</chatroom\_desc>

<member\_list>

<destination\_emp\_no>0407732</destination\_emp\_no>

<destination\_emp\_no>0407733</destination\_emp\_no>

<destination\_emp\_no>0407734</destination\_emp\_no>

</member\_list>

</LayoutHeader>

因為內文實質上是XML格式, 因此XML保留字元請先做HTMLENCODE

以下5個符號在http post之前需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: FlowER & IP Servers這樣的內文, 需轉換為FlowER &amp; IP Servers後再送出

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **資料名稱** | **母節點** | **出現次數** | **資料類型** | **描述** |
| group\_id | Content | 0-1 | string | 聊天室編號 |

Check Point:

增加qp\_chatroom.member欄位(nvarchar(250)), 如果request body的chatroom\_desc內容中group\_message=N, 代表是私聊, 創建完聊天室後, 需要將聊天室中兩人的工號寫進qp\_chatroom.member, ex:0407731,0305002

* 1. 當要新增聊天室前, 如果需要先檢查是否已經存在兩人的私聊, 也就是檢查qp\_chatroom.member, 已存在既有的工號組合, 則不新建聊天室, 取舊的聊天室groupid回傳給APP

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025918 | 新增聊天室失敗, 成員未註冊 | New chatroom failed, member not registered | 新增聊天室失败, 成员未注册 |
| 025919 | 傳入的成員不存在 | Chatroom member invalid | 传入的成员不存在 |
| 025926 | 保護名單必須是好友才能聊天 | You cannot invite the protected user who is not your friend | 保护名单必须是好友才能聊天 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_friend\_matrix**

**qp\_user**

**qp\_chatroom**

**\*\*\*後台實際找Jmessage更新的方式如下**

#### REQUEST URL

POST /v1/groups/

Example Request

{

"owner\_username": "tom",

"name": "群聊天室",

"members\_username": ["eddie", "annie"],

"desc": "运动"

}

Request Params

* owner\_username （必填）群主username
* name （必填）群组名字
  + 支持的字符：全部，包括 Emoji。
* members\_username 成员 username
* avatar （选填）群组头像，上传接口所获得的media\_id
* desc （选填） 群描述
  + 支持的字符：全部，包括 Emoji。

Example Response

< HTTP/1.1 201 Created

< Content-Type: application/json

<

{

"gid":12345,

"owner\_username": 123456,

"name": "display\_name",

"members\_username": [],

"desc":"doubi",

"MaxMemberCount" = 500,

"mtime" = "2014-07-01 00:00:00",

"ctime"="2014-06-05 00:00:00"

}

|  |  |  |
| --- | --- | --- |
| **Response参数** | **參數說明** | **字符长度限制** |
| name | 群组名称 | Byte(0~64) |
| desc | 群组描述 | Byte(0~250) |
| owner\_username | 群主的username | Byte(4-128) |
| MaxMemberCount | 群组默认500人 |  |
| ctime | 创建时间 |  |
| mtime | 最后修改时间 |  |
| avatar | 群组头像 |  |

### setQChatroom

#### 接口定義

透過此API可以修改聊天室標題及描述

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/setQChatroom?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 手機持有人的員工工號 |
| group\_id | LayoutHeader | Required | string | 聊天室編號 |
| chatroom\_name | LayoutHeader | Optional | string | 聊天室名稱 |
| chatroom\_desc | LayoutHeader | Optional | string | 聊天室描述, 用來存放給local讀取的資訊  ex:  是否需要歷史訊息  群聊或是私聊  定義如下:  need\_history=Y;group\_message=Y |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><group\_id>1000000</group\_id><chatroom\_name>聊天室名稱</chatroom\_name><chatroom\_desc>need\_history=Y;group\_message=Y</chatroom\_desc>

</LayoutHeader>

因為內文實質上是XML格式, 因此XML保留字元請先做HTMLENCODE

以下5個符號在http post之前需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: FlowER & IP Servers這樣的內文, 需轉換為FlowER &amp; IP Servers後再送出

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：NA**

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025903 | 必填欄位缺失 | Mandatory Field Lost | 必填字段缺失 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025920 | 傳入的聊天室編號無法識別 | The chatroom id is invalid | 传入的聊天室编号无法识别 |
| 025907 | 權限不足 | No Authority | 权限不足 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_user**

**qp\_chatroom**

**Chick Point:**

* + - * 1. 當使用者不存在此聊天室，不可修改聊天室資訊，return Error Code 025907 權限不足

**\*\*\*後台實際找Jmessage更新的方式如下**

#### REQUEST URL

PUT https://api.im.jpush.cn/v1/groups/{Group id}

Request Method (特別注意是https PUT)

**PUT** /v1/groups/{Group id}

Request Headers

(參數大小寫要一致)

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| Content-Type | 必填 | application/json |
| Authorization | 必填 | Basic base64(appKey:masterSecret)  其中 base64\_auth\_string 的生成算法为：base64(appKey:masterSecret) 即，对 appKey 加上冒号，加上 masterSecret 拼装起来的字符串，再做 base64 转换 |

Request Body

{ "desc" : "**need\_history=Y;group\_message=Y**" }

body參數說明:

* name 群名称
* desc 群描述
* need\_history, QChat自行定義的參數, 是否需要歷史訊息
* group\_message, QChat自行定義的參數, 是否為群聊

Example Response

HTTP/1.1 204 NO Content

**\*\*\*http回應碼為204, 如果已經正確更新的話, 回應204, 否則回應其他code**

### addQMember

#### 接口定義

透過此API可以新增成員

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/addQMember?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 新增聊天室人員的員工工號 |
| ~~lang~~ | ~~LayoutHeader~~ | ~~Required~~ | ~~string~~ | ~~ex:en-us~~ |
| ~~need\_push~~ | ~~LayoutHeader~~ | ~~Required~~ | ~~string~~ | ~~是否需要推播~~  ~~ex:Y, N~~ |
| ~~app\_key~~ | ~~LayoutHeader~~ | ~~Required~~ | ~~string~~ | ~~ex:appim~~ |
| group\_id | LayoutHeader | Required | string | 聊天室編號 |
| member\_list | LayoutHeader | Required | Container |  |
| destination\_emp\_no | member\_list | Required | string | 欲新增的聊天室成員  可以有多個  <destination\_emp\_no>0407732</destination\_emp\_no>  <destination\_emp\_no>0407733</destination\_emp\_no>  <destination\_emp\_no>0407734</destination\_emp\_no> |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><lang>en-us</lang><need\_push>Y</need\_push>

<app\_key>appim</app\_key>

<member\_list>

<destination\_emp\_no>0407732</destination\_emp\_no>

<destination\_emp\_no>0407733</destination\_emp\_no>

<destination\_emp\_no>0407734</destination\_emp\_no>

</member\_list>

</LayoutHeader>

因為內文實質上是XML格式, 因此XML保留字元請先做HTMLENCODE

以下5個符號在http post之前需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: FlowER & IP Servers這樣的內文, 需轉換為FlowER &amp; IP Servers後再送出

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：NA**

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025918 | 新增聊天室失敗, 成員未註冊 | New chatroom failed, member not registered | 新增聊天室失败, 成员未注册 |
| 025919 | 傳入的成員不存在 | Chatroom member invalid | 传入的成员不存在 |
| 025920 | 傳入的聊天室編號無法識別 | The chatroom id is invalid | 传入的聊天室编号无法识别 |
| 025926 | 保護名單必須是好友才能聊天 | You cannot invite the protected user who is not your friend | 保护名单必须是好友才能聊天 |
| 025929 | 私聊聊天室不可新增成員 | Private Chatroom can not add member | 私聊聊天室不可新增成员 |
| 025907 | 權限不足 | No Authority | 权限不足 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_friend\_matrix**

**qp\_user**

**qp\_chatroom**

**Check Point:**

* + - * 1. 當使用者不存在這個群組，不可加人，return Error code : 025907 權限不足

**\*\*\*後台實際找Jmessage更新的方式如下**

**\*\*\*請注意:批量增加与删除某 gid 群组的成员。群组成员将收到增加与删除成员的通知。**

#### REQUEST URL

POST https://api.im.jpush.cn/v1/groups/{Group id}/members

Request Method

**POST** https://api.im.jpush.cn/v1/groups/{Group id}/members

Request Headers

(參數大小寫要一致)

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| Content-Type | 必填 | application/json |
| Authorization | 必填 | Basic base64(appKey:masterSecret)  其中 base64\_auth\_string 的生成算法为：base64(appKey:masterSecret) 即，对 appKey 加上冒号，加上 masterSecret 拼装起来的字符串，再做 base64 转换 |

Request Body

{

"add":[

"Steven.Yan", "Cleo.W.Chan"

],

"remove":[

"Alan.Tu", "Darren.K.Ti"

]

}

body參數說明:

* 將要新增的人, 填在add區塊內
* 將要移出的人, 填在remove區塊內

如果只要新增, 不要移出, 則post body如下:

{

"add":[

"Steven.Yan", "Cleo.W.Chan"

],

"remove":[

]

}

Example Response

HTTP/1.1 204 NO Content

**\*\*\*http回應碼為204, 如果已經正確更新的話, 回應204, 否則回應其他code**

### removeQMember

#### 接口定義

透過此API可以移除成員

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/removeQMember?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 新增聊天室人員的員工工號 |
| ~~lang~~ | ~~LayoutHeader~~ | ~~Required~~ | ~~string~~ | ~~ex:en-us~~ |
| ~~need\_push~~ | ~~LayoutHeader~~ | ~~Required~~ | ~~string~~ | ~~是否需要推播~~  ~~ex:Y, N~~ |
| ~~app\_key~~ | ~~LayoutHeader~~ | ~~Required~~ | ~~string~~ | ~~ex:appim~~ |
| group\_id | LayoutHeader | Required | string | 聊天室編號 |
| member\_list | LayoutHeader | Required | Container |  |
| destination\_emp\_no | member\_list | Required | string | 欲移除的聊天室成員  可以有多個一次移除  <destination\_emp\_no>0407732</destination\_emp\_no>  <destination\_emp\_no>0407733</destination\_emp\_no>  <destination\_emp\_no>0407734</destination\_emp\_no> |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><lang>en-us</lang><need\_push>Y</need\_push>

<app\_key>appim</app\_key>

<member\_list>

<destination\_emp\_no>0407732</destination\_emp\_no>

<destination\_emp\_no>0407733</destination\_emp\_no>

<destination\_emp\_no>0407734</destination\_emp\_no>

</member\_list>

</LayoutHeader>

因為內文實質上是XML格式, 因此XML保留字元請先做HTMLENCODE

以下5個符號在http post之前需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: FlowER & IP Servers這樣的內文, 需轉換為FlowER &amp; IP Servers後再送出

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：NA**

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025918 | 新增聊天室失敗, 成員未註冊 | New chatroom failed, member not registered | 新增聊天室失败, 成员未注册 |
| 025919 | 傳入的成員不存在 | Chatroom member invalid | 传入的成员不存在 |
| 025920 | 傳入的聊天室編號無法識別 | The chatroom id is invalid | 传入的聊天室编号无法识别 |
| 025907 | 權限不足 | No Authority | 权限不足 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_friend\_matrix**

**qp\_user**

**qp\_chatroom**

**Chic Point**

1.當使用者不存在此聊天室，不可移除成員，return Error Code : 025907 權限不足

**\*\*\*後台實際找Jmessage更新的方式如下**

**\*\*\*請注意:批量增加与删除某 gid 群组的成员。群组成员将收到增加与删除成员的通知。**

#### REQUEST URL

POST https://api.im.jpush.cn/v1/groups/{Group id}/members

Request Method

**POST** https://api.im.jpush.cn/v1/groups/{Group id}/members

Request Headers

(參數大小寫要一致)

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| Content-Type | 必填 | application/json |
| Authorization | 必填 | Basic base64(appKey:masterSecret)  其中 base64\_auth\_string 的生成算法为：base64(appKey:masterSecret) 即，对 appKey 加上冒号，加上 masterSecret 拼装起来的字符串，再做 base64 转换 |

Request Body

{

"add":[

"Steven.Yan", "Cleo.W.Chan"

],

"remove":[

"Alan.Tu", "Darren.K.Ti"

]

}

body參數說明:

* 將要新增的人, 填在add區塊內
* 將要移出的人, 填在remove區塊內

如果只要移出, 不要新增, 則post body如下:

{

"add":[

],

"remove":[

"Alan.Tu", "Darren.K.Ti"

]

}

Example Response

HTTP/1.1 204 NO Content

**\*\*\*http回應碼為204, 如果已經正確更新的話, 回應204, 否則回應其他code**

### getQGroupHistoryMessage

#### 接口定義

透過此API可以從QPlay端獲得單一聊天室內的歷史訊息

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/getQGroupHistoryMessage?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 使用者的員工工號 |
| group\_id | LayoutHeader | Required | string | 聊天室編號 |
| begin\_time | LayoutHeader | Optional | string | unix時間搓記, 13位  要找尋的歷史訊息的起始時間, 單位到毫秒 |
| end\_time | LayoutHeader | Optional | string | unix時間搓記, 13位  要找尋的歷史訊息的結束時間, 單位到毫秒 |
| cursor | LayoutHeader | Optional | string | 選填  當放入cursor時表示要用cursor找尋從cursor發生的時間點之後的聊天訊息 |
| sort | LayoutHeader | Optional | string | 選填  0:create\_time從小排到大(預設)  1:create\_time從大牌到小 |

範例:

<LayoutHeader><emp\_no>0407731</emp\_no><group\_id>10122896</ group\_id><begin\_time> 1499999186000 </begin\_time><end\_time> 1499999286000</end\_time><cursor>B83D77056DD96A5AB890DB3960E1EA4D </cursor></LayoutHeader>

因為內文實質上是XML格式, 因此XML保留字元請先做HTMLENCODE

以下5個符號在http post之前需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: FlowER & IP Servers這樣的內文, 需轉換為FlowER &amp; IP Servers後再送出

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **資料名稱** | **母節點** | **出現次數** | **資料類型** | **描述** |
| total | Content | 1 | string | 此次查詢區間的總筆數 |
| cursor | Content | 1 | string | 這次查詢的指標, 若不是第一次獲取歷史資料的話, 可以用cursor查 |
| count | Content | 1 | string | 此次Json回傳的訊息筆數 |
| messages | Content | 1-N | container | 訊息container |
| set\_from\_name | messages | 0-1 | int | 目前固定是0 |
| from\_platform | messages | 0-1 | string | 发送方平台。  可选项： a - Android, i - iOS, w - WinPhone, web - Web |
| target\_name | messages | 0-1 | string | 接收者的展示名 |
| msg\_type | messages | 0-1 | string | 选项：text, voice, image, custom |
| version | messages | 0-1 | int | 协议版本号。第一版本：1，以此类推。 |
| target\_id | messages | 0-1 | string | 接收者ID。  可能值：${username}, ${gid}  接收方可用此字段，校验消息是不是发给自己的。  目前群聊, 只會有gid |
| sui\_mtime | messages | 0-1 | int |  |
| from\_appkey | messages | 0-1 | string | 發送端的Jpush app key  由 JPush Web Portal 生成的 24位字符串。字母或者数字，不区分大小写 |
| from\_name | messages | 0-1 | string | 发送方展示名。 |
| msg\_body | messages | 0-1 | container | body container |
| text | msg\_body | 0-1 | string | msg\_type =text時, 才有這個節點  顯示:聊天訊息 |
| height | msg\_body | 0-1 | int | msg\_type = image時, 才有這個節點  顯示:原图片调度。 |
| hash | msg\_body | 0-1 | string | msg\_type = image時, 才有這個節點  顯示:hash code |
| width | msg\_body | 0-1 | int | msg\_type = image時, 才有這個節點  顯示:原图片宽度。 |
| media\_id | msg\_body | 0-1 | string | msg\_type = image時, 才有這個節點  顯示:媒体文件上传到得到的KEY，用于生成下载URL。 |
| qplay\_media\_path | msg\_body | 0-1 | string | qplay server端保存圖片的URL路徑 |
| qplay\_thumb\_path | msg\_body | 0-1 | string | qplay server端保存圖片縮圖的URL路徑 |
| fsize | msg\_body | 0-1 | int | msg\_type = image時, 才有這個節點  顯示:文件大小（字节数） |
| media\_crc32 | msg\_body | 0-1 | int | msg\_type = image時, 才有這個節點  顯示:文件的 CRC32 校验码。 |
| from\_id | messages | 0-1 | string | 发送者 username |
| from\_type | messages | 0-1 | string | 发送方来源。  选项：user, robot, admin, ...。可用于扩展特定消息来源。  用户只允许发送 from\_type = user 的消息。 |
| create\_time | messages | 0-1 | int | 消息发送时间。  精确到秒 |
| target\_type | messages | 0-1 | string | 选项：single, group  群聊都是group |
| msgid | messages | 0-1 | int | 訊息主體的流水號, unique key |
| msg\_ctime | messages | 0-1 | int | // 服务器接收到消息的时间，单位毫秒 |
| msg\_level | messages | 0-1 | int | // 0代表应用内消息 1代表跨应用消息 |

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

**qp\_history**

**qp\_history\_file**

### getQGroupHistoryMessageJob (後台JOB專用)

#### 接口定義

透過此API可以從JMessage端獲得全部聊天室內的歷史訊息, 此API只有後台同步資料使用

#### 參數說明

1. API網址

https://report.im.jpush.cn/v2/messages?count=1000&begin\_time={begin\_time}&end\_time={end\_time}&cursor=${cursor}

1. 傳遞參數方法：

HTTPS GET

1. 網址列傳遞參數：

* count （必填）每次查询的总条数 一次最多1000
* begin\_time (必填) 记录开始时间 格式 yyyy-MM-dd HH:mm:ss 设置筛选条件大于等于begin time
* end\_time (必填) 记录结束时间 格式 yyyy-MM-dd HH:mm:ss 设置筛选条件下于等于end time
* begin\_time end\_time 之间最大范围不得超过7天
* cursor  当第一次请求后如果后面有数据，会返回一个cursor回来用这个获取接下来的消息 (cursor 有效时间是120s，过期后需要重第一个请求获取，重新遍历)
* 查询的消息按发送时间升序排序

**\*\*\*使用此接口，传递给JMessage的URL需要经过URL Encode处理，例如时间格式中的空格需要被转义为 %20**

1. Http request headers (參數大小寫要一致)

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| Content-Type | 必填 | application/json |
| Authorization | 必填 | Basic base64(appKey:masterSecret)  其中 base64\_auth\_string 的生成算法为：base64(appKey:masterSecret) 即，对 appKey 加上冒号，加上 masterSecret 拼装起来的字符串，再做 base64 转换 |

1. Http request body:

NA

1. Response body:

{

"total": 3000, "cursor":"APSK234ASDKQWE", "count": 1,

"messages": [

{ "target\_type": "single",

"msg\_type": "text",

"target\_name": "",

"target\_id": "10010648",

"from\_id": "868802000386631",

"from\_name": "868802000386631",

"from\_type": "user",

"from\_platform": "a",

"msg\_body": {

"text": "text",

"extras": { }

},

"create\_time": 1446016259,

"version": 1,

"msgid": 13242735,

"msg\_level" : 0, // 0代表应用内消息 1代表跨应用消息

"msg\_ctime" : 1466866468352 // 服务器接收到消息的时间，单位毫秒

}

]

}

**Response資料說明：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **資料名稱** | **母節點** | **出現次數** | **資料類型** | **描述** |
| total | Content | 1 | string | 此次查詢區間的總筆數 |
| cursor | Content | 1 | string | 這次查詢的指標, 在120sec內要重複查詢的話, 可以用cursor查  如果查詢區間的資料超過count參數的數量, 則可以用cursor查詢剩下的資料, 直到數量傳完為止 |
| count | Content | 1 | string | 此次Json回傳的訊息筆數 |
| messages | Content | 1-N | container | 訊息container |
| set\_from\_name | messages | 0-1 | int | 目前固定是0 |
| from\_platform | messages | 0-1 | string | 发送方平台。  可选项： a - Android, i - iOS, w - WinPhone, web - Web |
| target\_name | messages | 0-1 | string | 接收者的展示名 |
| msg\_type | messages | 0-1 | string | 选项：text, voice, image, custom |
| version | messages | 0-1 | int | 协议版本号。第一版本：1，以此类推。 |
| target\_id | messages | 0-1 | string | 接收者ID。  可能值：${username}, ${gid}  接收方可用此字段，校验消息是不是发给自己的。  目前群聊, 只會有gid |
| sui\_mtime | messages | 0-1 | int |  |
| from\_appkey | messages | 0-1 | string | 發送端的Jpush app key  由 JPush Web Portal 生成的 24位字符串。字母或者数字，不区分大小写 |
| from\_name | messages | 0-1 | string | 发送方展示名。 |
| msg\_body | messages | 0-1 | container | body container |
| text | msg\_body | 0-1 | string | msg\_type =text時, 才有這個節點  顯示:聊天訊息 |
| height | msg\_body | 0-1 | int | msg\_type = image時, 才有這個節點  顯示:原图片调度。 |
| hash | msg\_body | 0-1 | string | msg\_type = image時, 才有這個節點  顯示:hash code |
| width | msg\_body | 0-1 | int | msg\_type = image時, 才有這個節點  顯示:原图片宽度。 |
| media\_id | msg\_body | 0-1 | string | msg\_type = image時, 才有這個節點  顯示:媒体文件上传到得到的KEY，用于生成下载URL。 |
| fsize | msg\_body | 0-1 | int | msg\_type = image時, 才有這個節點  顯示:文件大小（字节数） |
| media\_crc32 | msg\_body | 0-1 | int | msg\_type = image時, 才有這個節點  顯示:文件的 CRC32 校验码。 |
| from\_id | messages | 0-1 | string | 发送者 username |
| from\_type | messages | 0-1 | string | 发送方来源。  选项：user, robot, admin, ...。可用于扩展特定消息来源。  用户只允许发送 from\_type = user 的消息。 |
| create\_time | messages | 0-1 | int | 消息发送时间。  精确到秒 |
| target\_type | messages | 0-1 | string | 选项：single, group  群聊都是group |
| msgid | messages | 0-1 | int | 訊息主體的流水號, unique key |
| msg\_ctime | messages | 0-1 | int | // 服务器接收到消息的时间，单位毫秒 |
| msg\_level | messages | 0-1 | int | // 0代表应用内消息 1代表跨应用消息 |

1. ResultCode說明:

JMessage 服务器端报的错误码。有可能出现在各平台的 SDK 里。

|  |  |  |
| --- | --- | --- |
| **Code** | **Error Message** | **说明** |
| 898000 | Server internal error | 内部错误 |
| 898001 | User exist | 用户已存在 |
| 898002 | No such user | 用户不存在 |
| 898003 | Parameter invalid! | 请求参数不合法 |
| 898004 | Password error | 更新密码操作，用户密码错误 |
| 898006 | Group id invalid | Group id不存在 |
| 898007 | Missing authen info | 校验信息为空 |
| 898008 | Basic authentication failed. | 校验失败 |
| 898009 | Appkey not exists | appkey不存在 |
| 898010 | Token expired | API请求 token 过期。正常情况下SDK会自动重新获取 token。 |
| 898011 | no auth to query other appkey's user or appkey no exist | 查询的appkey不具备跨应用权限 或者appkey不存在 |
| 898030 | Server response time out, please try again later | 系统繁忙，稍后重试 |
| 899000 | Server internal error | 系统内部错误 |
| 899001 | User exist | 用户已存在 |
| 899002 | No such user | 用户不存在 |
| 899003 | parameter invalid | 参数错误，Request Body参数不符合要求;resend 值不符合要求;用户名或者密码不合法;群组Group id不合法 |
| 899006 | Group id invalid | Group id 不存在 |
| 899007 | Missing authen info | 校验信息为空 |
| 899008 | Basic authentication failed | 校验失败 |
| 899009 | Appkey not exit | Appkey不存在 |
| 899011 | Repeat to add the members | 重复添加群成员 |
| 899012 | No enough space for members | 没有足够位置添加此次请求的成员 |
| 899013 | User list is bigger than 500 | 注册列表大于500，批量注册最大长度为500 |
| 899014 | User list is bigger than 500 | 添加操作操作成功 remove操作有username不存在讨论组中 remove失败 |
| 899015 | User 's group are full can not continue | 用户加入讨论组达到上限 |
| 899016 | No authority to send message | 用户没有管理员权限发送信息 |
| 899017 | There are usernames exist in blacklist | 用户已经被添加进黑名单 |
| 899018 | Admin can not be added into blacklist | 管理员不能被添加进黑名单 |
| 899019 | Here are usernames not exist in blacklist | 删除目标黑名单用户不存在黑名单中 |
| 899020 | no auth to operating other appkey | 跨应用失败 |
| 899021 | should use cross app api | 查询失败 应该使用跨应用api |
| 899043 | duplicate add user | 已经设置此用户为消息免打扰，重复设置错误 |
| 899044 | user is not exist in setting | 取消消息免打扰用户时，该用户不存在当前设置中 |
| 899045 | group is not exist | 设置群组消息免打扰时，群组不存在该系统中 |
| 899046 | user is not in group | 设置群组消息免打扰时，设置的群组，用户不在该群组中 |
| 899047 | duplicate add group | 已经设置此群组为消息免打扰，重复设置错误 |
| 899048 | already open global | 已经设置全局为消息免打扰，重复设置错误 |
| 899049 | group is not exist in setting | 取消消息免打扰群组时，该群组不存在当前设置中 |
| 899050 | already close global | 已经设置全局为消息免打扰，重复设置错误 |
| 899070 |  | 添加的好友已经存在好友列表中 |
| 899071 |  | 更新的好友不存在好友列表中 |
| 899030 | Server response time out, please try again later | 系统繁忙，稍后重试 |
| 800003 | appkey not exist | appkey未注册 |
| 800005 | user not exist | 用户ID未注册（appkey无该UID） |
| 800006 | user not exist | 用户ID不存在（数据库中无该UID） |
| 800008 | invalid request | 请求类型无法识别 |
| 800009 | system error | 服务器系统错误 |
| 800012 | user never login | 发起的用户处于登出状态，账号注册以后从未登录过，需要先登录 |
| 800013 | user logout | 发起的用户处于登出状态，请求的用户已经登出，需要先登录 |
| 800014 | appkey not match | 发起的用户appkey与目标不匹配 |
| 800016 | device not match | 发起的用户设备不匹配,当前请求的设备与上次登录的设备不匹配导致，需要先登录 |
| 801003 | user not exist | 登录的用户名未注册，登录失败 |
| 801004 | invalid password | 登录的用户密码错误，登录失败 |
| 801005 | invalid device | 登录的用户设备有误，登录失败 |
| 802002 | username not match | 登出用户名和登录用户名不匹配，登出失败 |
| 803001 | system error | 发送消息失败，状态存储异常 |
| 803002 | system error | 发送消息失败，系统网络异常 |
| 803003 | target user not exist | 发送消息失败，目标用户未注册或不存在 |
| 803004 | target group not exist | 发送消息失败，目标讨论组不存在 |
| 803005 | user not in group | 发送消息失败，发起者不在目标讨论组中 |
| 803006 | user not permitted | 发送消息失败，发起者权限不够或者类别不匹配 |
| 803007 | length of message exceed limit | 发送消息失败，消息长度超过限制 |
| 803008 | user in blacklist | 发送消息失败，发送者已被接收者拉入黑名单，仅限单聊 |
| 803009 | the message contains sensitive word: the word | 发送消息失败，消息内容包含敏感词汇：具体敏感词 |
| 803010 | beyond the frequency limit | 发送消息失败，发送频率超过系统限制，无法发送，客户端限制每分钟60条 |
| 803011 | msg content json error | 发送消息失败，消息格式配置错误 |
| 805001 | target user not exist | 目标用户不存在 |
| 805002 | already is friend | 添加好友失败：双方已经是好友 |
| 805003 | user not friend | 删除好友失败：目标用户并不是自己的好友 |
| 805004 | invalid friend memo | 修改好友备注失败：备注内容无效，无法修改 |
| 805006 | invitation event is not valid | 添加好友失败：邀请事件无效 |
| 808001 | group name invalid | 创建群组时群名为空，创建群组失败 |
| 808002 | user not permitted to create group | 用户无创建群组权限，创建群组失败 |
| 808003 | amount of group exceed limit | 用户拥有的群组数量已达上限,无法再创建 |
| 808004 | length of group name exceed limit | 群组名长度超出上限，创建群组失败 |
| 808005 | length of group desc exceed limit | 群组描述长度超出上限，创建群组失败 |
| 809001 | group id not exist | 目标群组ID不存在 |
| 809002 | user not in group | 用户不在该群组中 |
| 810001 | group id not exist | 目标群组ID不存在 |
| 810002 | add member list is null | 添加的成员列表为空 |
| 810003 | inviter not in group | 邀请方不在该群组中 |
| 810004 | user not permitted add member to group | 用户没有往群组中添加成员的权限 |
| 810005 | have member not register | 添加的成员列表中包含未注册成员 |
| 810007 | repeated added member | 添加的成员列表中有成员重复添加 |
| 810008 | amount of member exceed group limit | 添加的成员数量超出群组拥有的最大成员数上限 |
| 810009 | amount of group exceed member limit | 添加的成员列表中有成员拥有的群组数量已达上限 |
| 811001 | group id not exist | 目标群组ID不存在 |
| 811002 | del member list is null | 删除的成员列表为空 |
| 811003 | applicant not in group | 申请方不在该群组中 |
| 811004 | user not permitted delete member of group | 用户没有删除群组中成员的权限 |
| 811005 | have member not register | 删除的成员列表中有成员未注册 |
| 811006 | member of group not permitted deleted | 删除的成员列表中有成员该用户没有权限进行删除 |
| 811007 | repeated deleted member | 删除的成员列表中有成员重复删除 |
| 811008 | have member not in group | 删除的成员列表中有成员不在该群组中 |
| 812001 | group id not exist | 目标群组ID不存在 |
| 812002 | user not in group | 用户不在该群组中 |
| 812003 | length of group name exceed limit | 群组名超出长度上限 |
| 812004 | length of group desc exceed limit | 群组描述超出上限 |
| 818001 | zero member | 用户添加黑名单时，成员列表为空，添加失败 |
| 818002 | member not exist | 用户添加黑名单时，成员列表中有成员不存在，添加失败 |
| 818003 | member not permitted added | 用户添加黑名单时，成员列表中有成员不能被添加，添加失败 |
| 819001 | zero member | 用户移除好友出黑名单时，成员列表为空，操作失败 |
| 819002 | member not exist | 用户删除黑名单时，成员列表中有成员不存在，删除失败 |
| 831001 | member already set | 用户添加成员消息免打扰时，该成员已处于免打扰状态 |
| 832001 | member never set | 用户删除成员消息免打扰时，该成员不处于免打扰状态 |
| 833001 | group not exist | 用户添加群组消息免打扰时，该群组不存在 |
| 833002 | user not in group | 用户添加群组消息免打扰时，用户不存在该群组中 |
| 833003 | group already set | 用户添加群组消息免打扰时，该群组已处于免打扰状态 |
| 834001 | group never set | 用户删除群组消息免打扰时，该群组不处于免打扰状态 |
| 835001 | already set | 用户添加全局消息免打扰时，该用户已处于全局免打扰状态 |
| 836001 | never set | 用户删除全局消息免打扰时，该用户不处于全局免打扰状态 |
| 842001 | group not exist | 用户添加群组消息屏蔽时，该群组不存在 |
| 842002 | user not in group | 用户添加群组消息屏蔽时，用户不在该群组中 |
| 842003 | group already set | 用户添加群组消息屏蔽时，该群组已处于消息屏蔽状态 |
| 843001 | group never set | 用户删除群组消息屏蔽时，该群组不处于消息屏蔽状态 |
| 855001 | out of time | 消息撤回失败，超出撤回时间 |
| 855002 | request user is not message sender | 消息撤回失败，请求撤回方不是消息发送方 |
| 855003 | request message not exist | 消息撤回失败，请求撤回消息不存在 |
| 855004 | message already retract | 消息撤回失败，该消息已经撤回 |

**影響DB**

**qp\_history**

**qp\_history\_file**

**qp\_parameter**

**\*\*\*每次到Jmessage拿完資料後, 要更新en\_parameter.end\_time欄位**

**當下次的begin\_time使用**

**Cursor欄位不紀錄, 只用在每次迴圈拿Jmessage訊息時, 當變數使用**

範例:

透過https://report.im.jpush.cn/v2/groups/10122896/messages?count=500&begin\_time=2017-07-12%2010:15:00&end\_time=2017-07-18%2016:30:00

從JMessage端拿到的資訊

{

"total":17,

"cursor":"B83D77056DD96A5AB890DB3960E1EA4D",

"count":2,

"messages":[

{

"set\_from\_name":0,

"from\_platform":"i",

"target\_name":"539AF704-29BF-0DAC-06C4-BB5219003294",

"msg\_type":"text",

"version":1,

"target\_id":"10122896",

"sui\_mtime":1499999186,

"from\_appkey":"e343504d536ebce16b70167e",

"from\_name":"Test.K.SW",

"msg\_body":{

"text":"iOS1"

},

"from\_id":"Darren.K.Ti",

"from\_type":"user",

"create\_time":1500355702000,

"target\_type":"group",

"msgid":371251970,

"msg\_ctime":1500355702939,

"msg\_level":0

},

{"set\_from\_name":0,

"from\_platform":"a",

"target\_name":"539AF704-29BF-0DAC-06C4-BB5219003294",

"msg\_type":"image",

"version":1,

"target\_id":"10122896",

"from\_appkey":"e343504d536ebce16b70167e",

"from\_name":"Test.K.SW",

"from\_id":"Darren.K.Ti",

"msg\_body":{

"isFileUploaded":true,

"height":567,

"width":340,

"media\_id":"qiniu/image/a/23F9CB1A6D5345D07C6B657BAE1E0FE3",

"fsize":15371,

"media\_crc32":19437257,

"extras":{}

},

"create\_time":1500362478,

"from\_type":"user",

"target\_appkey":"",

"target\_type":"group",

"msgid":371341248,

"msg\_ctime":1500362481861,

"msg\_level":0

}

]

}

### getQUserChatroom

#### 接口定義

透過此API可以取得特定用戶所有聊天室資訊

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/ getQUserChatroom?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 手機持有人的員工工號 |

範例:

<LayoutHeader>

<emp\_no>1609009</emp\_no></LayoutHeader>

因為內文實質上是XML格式, 因此XML保留字元請先做HTMLENCODE

以下5個符號在http post之前需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: FlowER & IP Servers這樣的內文, 需轉換為FlowER &amp; IP Servers後再送出

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **資料名稱** | **母節點** | **出現次數** | **資料類型** | **描述** |
| chatroom\_list | Content | 0-1 | Container |  |
| desc | chatroom\_list | 0-N | string | 聊天室描述 |
| name | user\_list | 0-N | string | 聊天室名稱 |
| mtime | user\_list | 0-N | string | 最後修改時間 |
| gid | user\_list | 0-N | string | 聊天室id |
| avatar | user\_list | 0-N | String | 群組大頭貼 |
| ctime | user\_list | 0-N | String | 創建時間 |
| max\_member\_count | user\_list | 0-N | String | 群組人數上限(defult : 500) |

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

## 推播Token註冊

### sendPushToken

#### 接口定義

透過此API可以新增JMessage的push token

#### 參數說明

1. Custom API網址

*Custom API的網址為固定格式, 依據function name變化, 如下說明*

https://qplay.benq.com/qplayApi/public/v101/custom/appim/sendPushToken?lang=en-us&uuid=xxxxx

*後面則填上定義的參數, 其中lang及uuid為必填, QPlay要求輸入之必填項目*

1. 傳遞參數方法：

HTTPS POST

*Custom API只接受https post方式*

1. 網址列傳遞參數：

|  |  |  |  |
| --- | --- | --- | --- |
| **參數名稱** | **是否必填** | **資料類型** | **描述** |
| lang | Y | string | 語系, ex:en-us |
| uuid | Y | string | 手機的uuid識別碼 |

1. Http request headers: *(以下5個參數必填, 欄位名稱大小寫須跟以下範例一模一樣)*

|  |  |  |
| --- | --- | --- |
| 欄位名稱 | 是否必填 | 描述 |
| content-type | 必填 | 訊息體類型，類型需為application/json |
| app-key | 必填 | 專案名稱 |
| signature-time | 必填 | 產生Signature當下的時間(unix timestamp型式), 共10碼 |
| signature | 必填 | *請參考4.1章節說明, 將獲得的signature填到http headers內的signature欄位* |
| token | 必填 | 由server配發的token, 用來識別是否已經登入qplay  *token由QPlay app的Qlogin plugin取得* |

1. Http request body:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **參數名稱** | **父節點** | **是否必填** | **資料類型** | **描述** |
| LayoutHeader | NA | Required | Container |  |
| emp\_no | LayoutHeader | Required | string | 手機持有人的員工工號 |
| device\_type | LayoutHeader | Required | string | android  ios |
| push\_token | LayoutHeader | Required | string | im的推播token, 從jmessage的getRegisterID plugin取得 |

範例:

<LayoutHeader>

<emp\_no>1609009</emp\_no><device\_type>android</device\_type><push\_token>190e35f7e07040e1dc8</push\_token>

</LayoutHeader>

因為內文實質上是XML格式, 因此XML保留字元請先做HTMLENCODE

以下5個符號在http post之前需要先轉換為右邊格式

|  |  |
| --- | --- |
| 原始字元 | 轉換字元 |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quot; |

Ex: FlowER & IP Servers這樣的內文, 需轉換為FlowER &amp; IP Servers後再送出

1. Response body:

*QPlay收到來自App的request後, 直接將http request body也透過post方式轉送到外部系統的web service*

*收到回應資料後, 按以下格式提供回應訊息給app*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **節點標識** | **父節點標識** | **出現次數** | **資料類型** | **描述** |
| ResultCode | NA | 1 | Integer | 回應碼 |
| Message | NA | 1 | String | 回應訊息 |
| token\_valid | NA | 0-1 | Integer | 本次使用的token的有效時間  格式為Unix時間搓記  2016年3月25日 下午 01:49:56 換算為1458884996 |
| Content | NA | 0-1 | Container | 回應訊息內容Container |

**Content資料：NA**

1. ResultCode說明:

*custom API收到ResultCode後, 會依據多語系參數, 傳送正確的翻譯內容給app端, 規格書中, 請定義可能發生的result code*

*其中前三碼為project code, 請填入申請app-key時所產生的三碼project code*

*第四碼如果為9, 表示可以直接顯示在APP GUI介面上, 供user識別*

*若非9, 則表示為APP需自行處理的邏輯*

*例如:001013表示參數格式錯誤, 此error需APP端自行處理, 顯示給user也無事無補*

|  |  |  |  |
| --- | --- | --- | --- |
| **Result Code** | **Description - TW** | **Description - EN** | **Description - CN** |
| 1 | 成功 | OK | 成功 |
| 025908 | 帳號不存在 | Account Not Exist | 账号不存在 |
| 025915 | Content-Type錯誤 | Content type parameter invalid | Content-Type错误 |
| 025916 | 傳入的xml格式錯誤, Server端無法解析 | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | 傳入的json格式錯誤, Server端無法解析 | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025999 | 其他未知錯誤 | Unknown Error | 其他未知错误 |

**影響DB**

qp\_qchat\_push\_token

# JMessage Phone Gap v3.0.0

## 參數設定

|  |
| --- |
| var exec = require('cordova/exec') |
|  |  |
|  | var PLUGIN\_NAME = 'JMessagePlugin' |
|  |  |
|  | /\*\* |
|  | \* 针对消息发送动作的控制选项，可附加在消息发送方法的参数中。 |
|  | \*/ |
|  | var MessageSendingOptions = { |
|  | /\*\* |
|  | \* 接收方是否针对此次消息发送展示通知栏通知。 |
|  | \* @type {boolean} |
|  | \* @defaultvalue |
|  | \*/ |
|  | isShowNotification: true, |
|  | /\*\* |
|  | \* 是否让后台在对方不在线时保存这条离线消息，等到对方上线后再推送给对方。 |
|  | \* @type {boolean} |
|  | \* @defaultvalue |
|  | \*/ |
|  | isRetainOffline: true, |
|  | /\*\* |
|  | \* 是否开启了自定义接收方通知栏功能。 |
|  | \* @type {?boolean} |
|  | \*/ |
|  | isCustomNotificationEnabled: null, |
|  | /\*\* |
|  | \* 设置此条消息在接收方通知栏所展示通知的标题。 |
|  | \* @type {?string} |
|  | \*/ |
|  | notificationTitle: null, |
|  | /\*\* |
|  | \* 设置此条消息在接收方通知栏所展示通知的内容。 |
|  | \* @type {?string} |
|  | \*/ |
|  | notificationText: null |
|  | } |
|  |  |
|  | var EventHandlers = { |
|  | 'receiveMessage': [], |
|  | 'clickMessageNotification': [], |
|  | 'syncOfflineMessage': [], |
|  | 'syncRoamingMessage': [], |
|  | 'loginStateChanged': [], |
|  | 'contactNotify': [], |
|  | 'retractMessage': [] |
|  | } |
|  |  |
|  | var JMessagePlugin = { |
|  | /\*\* |
|  | \* @param {object} params = { |
|  | \* 'isOpenMessageRoaming': boolean // 是否开启消息漫游，不传默认关闭。 |
|  | \* } |
|  | \* |
|  | \* 打开消息漫游之后，用户多个设备之间登录时，SDK 会自动将当前登录用户的历史消息同步到本地。 |
|  | \*/ |
|  | init: function (params) { |
|  | var success = function (result) { |
|  | if (!EventHandlers.hasOwnProperty(result.eventName)) { |
|  | return |
|  | } |
|  |  |
|  | for (var index in EventHandlers[result.eventName]) { |
|  | EventHandlers[result.eventName][index].apply(undefined, [result.value]) |
|  | } |
|  | } |
|  |  |
|  | exec(success, null, PLUGIN\_NAME, 'init', [params]) |
|  | }, |
|  | /\*\* |
|  | \* 设置是否开启 debug 模式，开启后 SDK 将会输出更多日志信息。应用对外发布时应关闭。 |
|  | \* |
|  | \* @param {object} params = {'enable': boolean} |
|  | \*/ |
|  | setDebugMode: function (params) { |
|  | exec(null, null, PLUGIN\_NAME, 'setDebugMode', [params]) |
|  | }, |

## 登入:

|  |
| --- |
| /\*\* |
| \* @param {object} params = {'username': string, 'password': string} |
| \* @param {function} success = function () {} |
| \* @param {function} error = function ({'code': '错误码', 'description': '错误信息'}) {} |
| \*/ |
| login: function (params, success, error) { |
| exec(success, error, PLUGIN\_NAME, 'userLogin', [params]) |
| }, |

## 登出:

|  |
| --- |
| /\*\* |
| \* 用户登出接口，调用后用户将无法收到消息。登出动作必定成功，开发者不需要关心结果回调。 |
| \* |
| \* @param {function} success = function () {} |
| \*/ |
| logout: function () { |
| exec(null, null, PLUGIN\_NAME, 'userLogout', []) |
|  |

## 獲取自己帳號訊息:

|  |
| --- |
| /\*\* |
| \* 登录成功则返回用户信息，已登出或未登录则对应用户信息为空对象。 |
| \* |
| \* @param {function} success = function (myInfo) {} |
| \*/ |
| getMyInfo: function (success) { |
| exec(success, null, PLUGIN\_NAME, 'getMyInfo', []) |
| }, |

## 獲取用戶訊息:

|  |
| --- |
| /\*\* |
| \* 获取用户信息，此接口可用来获取不同 appKey 下用户的信息，如果 appKey 为空，则默认获取当前 appKey 下的用户信息。 |
| \* |
| \* @param {object} params = {'username': string, 'appKey': string} |
| \* @param {function} success = function (userInfo) {} // 通过参数返回用户对象。 |
| \* @param {function} error = function ({'code': '错误码', 'description': '错误信息'}) {} |
| \*/ |
| getUserInfo: function (params, success, error) { |
| exec(success, error, PLUGIN\_NAME, 'getUserInfo', [params]) |
| }, |

## 送出文字訊息:

|  |
| --- |
| /\*\* |
| \* @param {object} params = { |
| \* 'type': string, // 'single' / 'group' |
| \* 'groupId': string, // 当 type = group 时，groupId 不能为空 |
| \* 'username': string, // 当 type = single 时，username 不能为空 |
| \* 'appKey': string, // 当 type = single 时，用于指定对象所属应用的 appKey。如果为空，默认为当前应用 |
| \* 'text': string, // 消息内容 |
| \* 'extras': object, // Optional. 自定义键值对 = {'key1': 'value1'} |
| \* 'messageSendingOptions': MessageSendingOptions // Optional. MessageSendingOptions 对象 |
| \* } |
| \* @param {function} success = function (msg) {} // 以参数形式返回消息对象。 |
| \* @param {function} error = function ({'code': '错误码', 'description': '错误信息'}) {} |
| \*/ |
| sendTextMessage: function (params, success, error) { |
| exec(success, error, PLUGIN\_NAME, 'sendTextMessage', [params]) |
| }, |

## 送出圖片訊息:

|  |
| --- |
| /\*\* |
| \* @param {object} params = { |
| \* 'type': string, // 'single' / 'group' |
| \* 'groupId': string, // 当 type = group 时，groupId 不能为空 |
| \* 'username': string, // 当 type = single 时，username 不能为空 |
| \* 'appKey': string, // 当 type = single 时，用于指定对象所属应用的 appKey。如果为空，默认为当前应用。 |
| \* 'path': string, // 本地图片绝对路径，类似：/storage/emulated/0/DCIM/Camera/IMG\_20160526\_130223.jpg |
| \* 'extras': object, // Optional. 自定义键值对 = {'key1': 'value1'} |
| \* 'messageSendingOptions': MessageSendingOptions // Optional. MessageSendingOptions 对象 |
| \* } |
| \* @param {function} success = function (msg) {} // 以参数形式返回消息对象。 |
| \* @param {function} error = function ({'code': '错误码', 'description': '错误信息'}) {} |
| \*/ |
| sendImageMessage: function (params, success, error) { |
| exec(success, error, PLUGIN\_NAME, 'sendImageMessage', [params]) |
| }, |

## 獲取歷史訊息:

|  |
| --- |
| /\*\* |
| \* 从最新的消息开始获取历史消息。 |
| \* 当 from = 0 && limit = =1 时，返回所有历史消息。 |
| \* |
| \* @param {object} params = { |
| \* 'type': string, // 'single' / 'group' |
| \* 'groupId': string, // 当 type = group 时，groupId 不能为空。 |
| \* 'username': string, // 当 type = single 时，username 不能为空。 |
| \* 'appKey': string, // 当 type = single 时，用于指定对象所属应用的 appKey。如果为空，默认为当前应用。 |
| \* 'from': Number, // 开始的消息下标。 |
| \* 'limit': Number // 要获取的消息数。比如当 from = 0, limit = 10 时，是获取第 0 - 9 的 10 条历史消息。 |
| \* } |
| \* @param {function} success = function (messageArray)) {} // 以参数形式返回历史消息对象数组 |
| \* @param {function} error = function ({'code': '错误码', 'description': '错误信息'}) {} |
| \*/ |
| getHistoryMessages: function (params, success, error) { |
| exec(success, error, PLUGIN\_NAME, 'getHistoryMessages', [params]) |
| }, |

## 獲取當前USER所有的聊天室ID:

|  |
| --- |
| /\*\* |
| \* 获取当前用户所有所在的群组 id。 |
| \* |
| \* @param {function} success = function (groupIdArray) {} // 以参数形式返回 group id 数组 |
| \* @param {function} error = function ({'code': '错误码', 'description': '错误信息'}) {} |
| \*/ |
| getGroupIds: function (success, error) { |
| exec(success, error, PLUGIN\_NAME, 'getGroupIds', []) |
| }, |

## 獲取聊天室基本訊息:

|  |
| --- |
| /\*\* |
| \* @param {object} params = {'id': '群组 id'} |
| \* @param {function} success = function (groupInfo) {} // 以参数形式返回群组信息对象 |
| \* @param {function} error = function ({'code': '错误码', 'description': '错误信息'}) {} |
| \*/ |
| getGroupInfo: function (params, success, error) { |
| exec(success, error, PLUGIN\_NAME, 'getGroupInfo', [params]) |
| }, |

## 退出聊天室:

|  |
| --- |
| /\*\* |
| \* @param {object} params = {'id': '群组 id'} |
| \* @param {function} success = function () {} |
| \* @param {function} error = function ({'code': '错误码', 'description': '错误信息'}) {} |
| \*/ |
| exitGroup: function (params, success, error) { |
| exec(success, error, PLUGIN\_NAME, 'exitGroup', [params]) |
| }, |

## 獲取聊天室成員:

|  |
| --- |
| /\*\* |
| \* @param {object} params = {'id': '群组 id'} |
| \* @param {function} success = function (userInfoArray) {} // 以参数形式返回用户对象数组 |
| \* @param {function} error = function ({'code': '错误码', 'description': '错误信息'}) {} |
| \*/ |
| getGroupMembers: function (params, success, error) { |
| exec(success, error, PLUGIN\_NAME, 'getGroupMembers', [params]) |
| }, |

## 下載指定消息的原圖:

|  |
| --- |
| /\*\* |
| \* 下载指定图片消息的原图，如果已经下载，会直接返回本地文件路径，不会重复下载。 |
| \* |
| \* @param {object} params = { |
| \* 'type': string, // 'single' / 'group' |
| \* 'groupId': string, // 目标群组 id。 |
| \* 'username': string, // 目标用户名。 |
| \* 'appKey': string, // 目标用户所属 AppKey。 |
| \* 'messageId': string // 指定消息 id。 |
| \* } |
| \* @param {function} success = function ({'messageId': string, 'filePath': string}) {} |
| \* @param {function} error = function ({'code': '错误码', 'description': '错误信息'}) {} |
| \*/ |
| downloadOriginalImage: function (params, success, error) { |
| exec(success, error, PLUGIN\_NAME, 'downloadOriginalImage', [params]) |
| }, |

## 進入聊天對話:

|  |
| --- |
| /\*\* |
| \* 进入聊天会话。可以在进入聊天会话页面时调用该方法，这样在收到当前聊天用户的消息时，不会显示通知。 |
| \* |
| \* @param {object} params = { |
| \* 'type': string, // 'single' / 'group' |
| \* 'groupId': string, // 目标群组 id。 |
| \* 'username': string, // 目标用户名。 |
| \* 'appKey': string, // 目标用户所属 AppKey。 |
| \* } |
| \* @param {function} success = function () {} |
| \* @param {function} error = function ({'code': '错误码', 'description': '错误信息'}) {} |
| \*/ |
| enterConversation: function (params, success, error) { |
| if (device.platform === 'Android') { |
| exec(success, error, PLUGIN\_NAME, 'enterConversation', [params]) |
| } |
| }, |

## 離開聊天對話:

|  |
| --- |
| exitConversation: function () { |
| if (device.platform === 'Android') { |
| exec(null, null, PLUGIN\_NAME, 'exitConversation', []) |
| } |
| }, |

## 獲取單一個聊天對話:

|  |
| --- |
| /\*\* |
| \* @param {object} params = { |
| \* 'type': string, // 'single' / 'group' |
| \* 'groupId': string, // 目标群组 id。 |
| \* 'username': string, // 目标用户名。 |
| \* 'appKey': string, // 目标用户所属 AppKey。 |
| \* } |
| \* @param {function} success = function (conversation) {} // 以参数形式返回聊天会话对象。 |
| \* @param {function} error = function ({'code': '错误码', 'description': '错误信息'}) {} |
| \*/ |
| getConversation: function (params, success, error) { |
| exec(success, error, PLUGIN\_NAME, 'getConversation', [params]) |
| }, |

## 獲取所有聊天對話:

|  |
| --- |
| /\*\* |
| \* @param {function} success = function (conversationArray) {} // 以参数形式返回会话对象数组。 |
| \* @param {function} error = function ({'code': '错误码', 'description': '错误信息'}) {} |
| \*/ |
| getConversations: function (success, error) { |
| exec(success, error, PLUGIN\_NAME, 'getConversations', []) |
| }, |

## 重置單一個聊天對話的未讀消息數

|  |
| --- |
| /\*\* |
| \* 重置单个会话的未读消息数。 |
| \* |
| \* @param {object} params = { |
| \* 'type': string, // 'single' / 'group' |
| \* 'groupId': string, // 目标群组 id。 |
| \* 'username': string, // 目标用户名。 |
| \* 'appKey': string, // 目标用户所属 AppKey。 |
| \* } |
| \* @param {function} success = function () {} |
| \* @param {function} error = function ({'code': '错误码', 'description': '错误信息'}) {} |
| \*/ |
| resetUnreadMessageCount: function (params, success, error) { |
| exec(success, error, PLUGIN\_NAME, 'resetUnreadMessageCount', [params]) |
| }, |

## 監聽訊息類的事件:

|  |
| --- |
| /\*\* |
| \* 添加收到消息事件监听。 |
| \* |
| \* @param {function} listener = function (message) {} // 以参数形式返回消息对象。 |
| \* message = { |
| \* 'id': string, |
| \* 'from': object, // 消息发送者信息对象。 |
| \* 'target': object, // 消息接收方信息（可能为用户或者群组）。 |
| \* 'type': string // 'text' / 'image' / 'voice' / 'location' / 'file' / 'custom' / 'event' |
| \* ... // 不同消息类型还有其他对应的相关字段，具体可参考文档。 |
| \* } |
| \*/ |
| addReceiveMessageListener: function (listener) { |
| EventHandlers.receiveMessage.push(listener) |
| }, |
| removeReceiveMessageListener: function (listener) { |
| var handlerIndex = EventHandlers.receiveMessage.indexOf(listener) |
| if (handlerIndex >= 0) { |
| EventHandlers.receiveMessage.splice(handlerIndex, 1) |
| } |
| }, |

## 監聽事件類的事件:

|  |
| --- |
| /\*\* |
| \* 添加点击通知栏消息通知事件监听。 |
| \* Note: Android only, (如果想要 iOS 端 实现相同的功能，需要同时集成 jpush-phonegap-plugin) |
| \* @param {function} listener = function (message) {} // 以参数形式返回消息对象。 |
| \*/ |
| addClickMessageNotificationListener: function (listener) { |
| EventHandlers.clickMessageNotification.push(listener) |
| }, |
| removeClickMessageNotificationListener: function (listener) { |
| var handlerIndex = EventHandlers.clickMessageNotification.indexOf(listener) |
| if (handlerIndex >= 0) { |
| EventHandlers.clickMessageNotification.splice(handlerIndex, 1) |
| } |
| }, |

## 監聽離線消息:

|  |
| --- |
| /\*\* |
| \* 添加同步离线消息事件监听。 |
| \* |
| \* @param {function} listener = function ({'conversation': {}, 'messageArray': []}) {} // 以参数形式返回消息对象数组。 |
| \*/ |
| addSyncOfflineMessageListener: function (listener) { |
| EventHandlers.syncOfflineMessage.push(listener) |
| }, |
| removeSyncOfflineMessageListener: function (listener) { |
| var handlerIndex = EventHandlers.syncOfflineMessage.indexOf(listener) |
| if (handlerIndex >= 0) { |
| EventHandlers.syncOfflineMessage.splice(handlerIndex, 1) |
| } |
| }, |

## 監聽登入狀態:

|  |
| --- |
| /\*\* |
| \* 添加登录状态变更事件监听。 |
| \* |
| \* @param {function} listener = function (event) {} // 以参数形式返回事件信息。 |
| \* event = { |
| \* 'type': string, // 'user\_password\_change' / 'user\_logout' / 'user\_deleted' / 'user\_login\_status\_unexpected' |
| \* } |
| \*/ |
| addLoginStateChangedListener: function (listener) { |
| EventHandlers.loginStateChanged.push(listener) |
| }, |
| removeLoginStateChangedListener: function (listener) { |
| var handlerIndex = EventHandlers.loginStateChanged.indexOf(listener) |
| if (handlerIndex >= 0) { |
| EventHandlers.loginStateChanged.splice(handlerIndex, 1) |
| } |
| }, |

## 返回物件的元素組成:

UserInfo: {

type: 'user',

username: string, // 用户名。

appKey: string, // 用户所属应用的 appKey。可与 username 共同作为用户的唯一标志。

nickname: string, // 昵称。

gender: string, // 'male' / 'female' / 'unknown'

avatarThumbPath: string, // 头像的缩略图地址。

birthday: number, // 日期的毫秒数。

region: string, // 地区。

signature: string, // 个性签名。

address: string, // 具体地址。

noteName: string, // 备注名。

noteText: string, // 备注信息。

isNoDisturb: boolean, // 是否免打扰。

isInBlackList: boolean, // 是否在黑名单中。

isFriend:boolean // 是否为好友。

}

GroupInfo: {

type: 'group',

id: string, // 群组 id，

name: string, // 群组名称。

desc: string, // 群组描述。

level: number, // 群组等级，默认等级 4。

owner: string, // 群主的 username。

ownerAppKey: string, // 群主的 appKey。

maxMemberCount: number, // 最大成员数。

isNoDisturb: boolean, // 是否免打扰。

isBlocked: boolean // 是否屏蔽群消息。

}

Conversation: {

/\*\*

\* 会话对象标题。

\* 如果为群聊：

\* - 未设置群名称：自动使用群成员中前五个人的名称拼接成 title。

\* - 设置了群名称，则显示群名称。

\* 如果为单聊：如果用户有昵称，显示昵称。否则显示 username。

\*/

title: string,

latestMessage: Message, // 最近的一条消息对象。

unreadCount: number, // 未读消息数。

conversationType: 'single' / 'group',

target: UserInfo / GroupInfo // 聊天对象信息。

}

## Message元素

TextMessage: {

id: string, // 消息 id。

type: 'text', // 消息类型。

from: UserInfo, // 消息发送者对象。

target: UserInfo / GroupInfo, // 消息接收者对象。可能是用户或群组。

createTime: number, // 发送消息时间。

text: string, // 消息内容。

extras: object // 附带的键值对对象。

}

ImageMessage: {

id: string,

type: 'image',

from: UserInfo,

target: UserInfo / GroupInfo,

extras: object,

thumbPath: string // 图片的缩略图路径。要下载原图需要调用 `downloadOriginalImage` 方法。

}

VoiceMessage: {

id: string,

type: 'image',

from: UserInfo,

target: UserInfo / GroupInfo,

extras: object,

path: string, // 语音文件路径。

duration: number // 语音时长

}

LocationMessage: {

id: string,

type: 'voice',

from: UserInfo,

target: UserInfo / GroupInfo,

extras: object,

address: string, // 详细地址。

longitude: number, // 经度。

latitude: number, // 纬度。

scale:number // 地图缩放比例。

}

FileMessage: {

id: string,

type: 'file',

from: UserInfo,

target: UserInfo / GroupInfo,

extras: object,

fileName: string // 文件名。要下载完整文件需要调用 `downloadFile` 方法。

}

CustomMessage: {

id: string,

type: 'file',

from: UserInfo,

target: UserInfo / GroupInfo,

extras: object,

customObject: object // 自定义键值对对象。

}

## Event元素:

Event: {

type: 'event',

eventType: string, // 'group\_member\_added' / 'group\_member\_removed' / 'group\_member\_exit'

usernames: Array // 该事件涉及到的用户 username 数组。

}

# Error Code List

以下說明Error Code的定義

QPlay類的錯誤碼, 編號為999開頭, QPlay回應999開頭的六位數Error Code表示為程式面錯誤, 不需要錯誤消息呈現在APP上, 開發過程中看到999類的訊息, 應直接fix bug

*請先列出以下所有999類的訊息給RD, 以下訊息都需要fix bug*

|  |  |
| --- | --- |
| **Result Code** | **Description** |
| 1 | 成功 |
| 999001 | 傳入參數不足或傳入參數格式錯誤 |
| 999002 | version參數錯誤 or 不存在 |
| 999003 | action參數錯誤 or 不存在 |
| 999004 | lang參數錯誤 or 不存在 |
| 999005 | 傳送方法錯誤,  ex.Method應為GET時, 卻傳POST |
| 999006 | Content-Type錯誤  (當Client的傳送方法為POST時, 才會有此result code產生) |
| 999007 | 傳入的json格式錯誤, Server端無法解析  (當Client的傳送方法為POST時, 才會有此result code產生) |
| 999008 | Signature驗證碼不正確 |
| 999009 | 禁止存取API (IP為黑名單) |
| 999010 | app-key參數錯誤 |
| 999011 | signature參數錯誤或誤差超過15分鐘 |
| 999012 | app已經下架 |
| 999013 | push token已經使用 |
| 999014 | company不存在 |
| 999015 | APP未上傳 |
| 999016 | 調用custom API參數錯誤 |
| 999999 | 其他未知錯誤 |

QChat專案所會使用的回應代碼前三碼為025

詳細清單列出於下

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Result Code** | | **Description - TW** | | **Description - EN** | **Description - CN** |
| 1 | | 成功 | | OK | 成功 |
| 025901 | | 成功 | | OK | 成功 |
| 025902 | | Location或是Function錯誤 | | Location or Function Not Found | Location或是Function错误 |
| 025903 | | 必填欄位缺失 | | Mandatory Field Lost | 必填字段缺失 |
| 025904 | | 查無事件資料 | | No Data | 查无事件资料 |
| 025905 | | 欄位格式錯誤 | | Field Format Error | 字段格式错误 |
| 025906 | | 查無分類基本資料 | | No Category Information | 查无分类基本数据 |
| 025907 | 權限不足 | | No Authority | | 权限不足 |
| 025908 | | 帳號不存在 | | Account Not Exist | 账号不存在 |
| 025909 | | 查無Task資料 | | No Data | 查无任務资料 |
| 025910 | | 無法編輯已完成事件 | | Can not edit closed event | 无法编辑已完成事件 |
| 025911 | | 關聯事件狀態異常 | | Related event status error | 关联事件状态异常 |
| 025912 | | 事件類型錯誤 | | Event type error | 事件类型错误 |
| 025913 | | 事件狀態碼錯誤 | | Event status code error | 事件状态码错误 |
| 025914 | | 任務狀態碼錯誤 | | Task status code error | 任務状态码错误 |
| 025915 | | Content-Type錯誤 | | Content type parameter invalid | Content-Type错误 |
| 025916 | | 傳入的xml格式錯誤, Server端無法解析 | | Input Xml format is invalid | 传入的xml格式错误, Server端无法解析 |
| 025917 | | 傳入的json格式錯誤, Server端無法解析 | | Input Json format is invalid | 传入的json格式错误, Server端无法解析 |
| 025918 | | 新增聊天室失敗, 成員未註冊 | | New chatroom failed, member not registered | 新增聊天室失败, 成员未注册 |
| 025919 | | 傳入的成員不存在 | | Chatroom member invalid | 传入的成员不存在 |
| 025920 | | 傳入的聊天室編號無法識別 | | The chatroom id is invalid | 传入的聊天室编号无法识别 |
| 025921 | | 要設定的好友工號不存在 | | The destination employee number is invalid | 要设定的好友工号不存在 |
| 025922 | | 要設定的好友是保護名單 | | The destination employee number is protect user | 要设定的好友是保护名单 |
| 025923 | | 要設定對象已經是好友 | | You cannot send the invitation to friends. | 要设定对象已经是好友 |
| 025924 | | 要邀請的好友已經註冊過QPlay | | The destination employee already registered | 要邀请的好友已经注册过QPlay |
| 025925 | | 不得修改他人資料 | | Cannot modify other people’s basic information | 不得修改他人数据 |
| 025926 | | 保護名單必須是好友才能聊天 | | You cannot invite the protected user who is not your friend | 保护名单必须是好友才能聊天 |
| 025927 | | 已發送過交友邀請 | | Invitation already send | 已发送过交友邀请 |
| 025928 | | 交友邀請不存在 | | Invitation not exist | 交友邀请不存在 |
| 025929 | | 私聊聊天室不可新增成員 | | Private Chatroom can not add member | 私聊聊天室不可新增成员 |
| 025930 | | 調用極光 API出错 | | Call API FailedOrErrorOccurs | 调用极光API出错 |
| 025998 | | 查無資料 | | No Data | 查无資料 |
| 025999 | | 其他未知錯誤 | | Unknown Error | 其他未知错误 |

# Others